AD-A215 739

(3)

RADC-TR-89-209, Vol III (of three)
Final Technical Report
October 1989

COMPUTER-AIDED DESIGN FOR BUILT-IN-TEST (CADBIT) - Software Specification

Grumman Aerospace Corporation

Tracy Kelly

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

89 12 [illegible]

RADC-TR-89-209, Vol III (of three) has been reviewed and is approved for publication.

APPROVED: [illegible]

JAMES J. VACCARO
Project Engineer

APPROVED: John J. Bart

JOHN J. BART
Technical Director
Directorate of Reliability & Compatibility

FOR THE COMMANDER: [illegible]

JAMES W. HYDE III
Directorate of Plans & Programs

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS: N/A
2a. SECURITY CLASSIFICATION AUTHORITY: N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): N/A	5. MONITORING ORGANIZATION REPORT NUMBER(S): RADC-TR-89-209, Vol III (of three)
6a. NAME OF PERFORMING ORGANIZATION: Grumman Aerospace Corporation; 6b. OFFICE SYMBOL (If applicable):	7a. NAME OF MONITORING ORGANIZATION: Rome Air Development Center (RBES)
6c. ADDRESS (City, State, and ZIP Code): South Oyster Bay Rd, Bethpage NY 11714	7b. ADDRESS (City, State, and ZIP Code): Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Rome Air Development Center; 8b. OFFICE SYMBOL (If applicable): RBES	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: F30602-86-C-0219
8c. ADDRESS (City, State, and ZIP Code): Griffiss AFB NY 13441-5700	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. 62702F; PROJECT NO. 2940; TASK NO 01; WORK UNIT ACCESSION NO 01

11. TITLE (Include Security Classification)
COMPUTER-AIDED DESIGN FOR BUILT-IN-TEST (CADBIT) - Software Specification

12. PERSONAL AUTHOR(S)
Tracy Kelly

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Sep 86 TO Sep 88	October 1989	154

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
12	05	
14	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Built-In-Test; CADBIT Software; Computer Aided Design; BIT data base; Printed Circuit Boards

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
The Computer Aided Design for Built-In-Test (CADBIT) Final Report consists of three volumes organized as follows. Volume I is a general description including introduction, automated procedure, data base, menus, CAD and BIT survey, and recommendations. Volume II contains a description of the BIT data base library element and BIT library elements for 13 BIT techniques, which were found to be suitable for CADBIT. Volume III contains the CADBIT software requirements specification to be used as a basis for encoding the CADBIT software modules and the creation of its data base.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: James M. Vaccaro	22b. TELEPHONE (Include Area Code): (315) 330-4205; 22c. OFFICE SYMBOL: RADC (RBES)

DD Form 1473, JUN 86 — Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

APPENDICES

1.0 SUMMARY

This report is one of three volumes. The Executive Summary describes the entire CAD-BIT effort and the following Volume Summary is a description of this volume.

1.1 EXECUTIVE SUMMARY

CAD-BIT is a development program to specify the implementation of an automated procedure to integrate Built-In-Test (BIT) into the design of Printed Circuit Boards (PCBs) or Computer-Aided Design (CAD) workstations. When fully developed, the CAD-BIT software will be capable of operating on generic workstations which meet various standards. These standards include those for operating system (UNIX), programming language (C), and graphical data interchange Initial Graphics Exchange Specification (IGES).

The purpose of this program was to develop the design of the automated procedure, the associated BIT data base, and a software specification for the CAD-BIT module ready for encoding. No coding of the CAD-BIT Module (CBM) was performed except as necessary to test and verify feasibility. CAD workstations and BIT techniques and their applications were also surveyed to determine standards required for the CAD-BIT module implementation and to establish requirements for and define the structure of the BIT data base.

1.1.1 SCOPE

This report describes the development of the CAD-BIT automated procedure, the associated Data Base of BIT Functions, and software specification developed during this contract. The contents of this report are organized into the three volumes described below.

Volume I Technical Issues

Volume I is a general CAD-BIT description and provides useful information for any type of involvement with CAD-BIT. It begins with an Executive Summary describing the work performed under the CAD-BIT contract. It is followed by a detailed description of the automated procedure. The description contains text, flow diagrams of the procedure operations and its data, sets of menu sequences showing menu options, selections, and resulting operations. Algorithms and formulas are included. The CAD-BIT Data Base and its files are described.

Additional topics in Volume I include Menus, the CAD-BIT Feasibility Demonstration, BIT and CAD workstation surveys and Standards Recommendations, SMART-BIT Applications, and a Automated Procedure Evaluation. The Volume also includes an appendix with a BIT library example for the On-Board ROM BIT Technique.

Volume II BIT Library

Volume II contains a description of the BIT data base library elements and BIT library elements for the thirteen BIT techniques listed below. The data in Volume II will be used to encode CAD-BIT's BIT technique data base during the implementation phase. In addition, it illustrates the required data for adding new BIT techniques. It also provides useful data to the future circuit designer / CAD-BIT user on the BIT techniques, their implementation, and the default circuit components.

- On-Board ROM
- Microprocessor BIT
- Microdiagnostics
- On-Board Integration of VLSI Chips BIT (OBIVCB)
- Built-In Logic Block Observer (BILBO)
- Error Detection and Correction Codes
- Scan
- Digital Wraparound
- Pseudo Random Pattern Generator with Multiple Input Shift Register (PRPG/MISR)
- Comparator
- Voltage Summing
- Redundancy
- Analog Wraparound

[illegible]

Volume III CAD-BIT Software Specification

[illegible]

Volume III contains the CAD-BIT Software Requirements Specification ~~(SRS)~~ which establishes the requirements for the Computer Software Configuration Item ~~(CSCI)~~ identified as ~~Computer-Aided~~ Design for Built-In-Test ~~(CAD-BIT)~~ System. It will be used during the implementation as the basis for encoding the CAD-BIT software modules and the creation of its data base. Computer aided design, computer aided manufacturing (FDC)

1.1.2 PURPOSE

The purpose of the CAD-BIT system is to provide an automated procedure to aid the electronic circuit designer in the selection of BIT techniques, the insertion of the associated BIT circuitry into the PCB design, and to provide a post design evaluation of the penalties incurred by the addition of BIT circuitry into the PCB functional design.

1.2 VOLUME III SUMMARY

This Specification defines the features of the CAD-BIT automated procedure. In the subparagraphs which follow, both a brief overview as well as a structured overview of the CAD-BIT procedure is provided. Illustrations of the overviews are also provided in Figures 1-1 and 1-2.

1.2.1 CAD-BIT PROCEDURE OVERVIEW

A PCB functional design is entered into the Computer Aided Design (CAD) system in the form of an electrical schematic or logic diagram. During this and all CAD-BIT phases, all host system functions (CAD and other) are available to the designer who is an electrical circuit design engineer and is familiar with the operation and capabilities of the CAD host system.

During the Selection process the designer generates a User Design Profile which describes, in terms of attributes, the PCB (or a partition of that PCB) being designed. The design can be partitioned so that CAD-BIT can be applied repeatedly on different portions of the total design. This enables several BIT techniques to be applied to the PCB design and any technique to be used repeatedly. Each separate group of BIT circuitry covering a portion of the functional circuitry is considered a separate BIT Group. As part of the profile, different penalty attribute weights can be applied to each BIT group.

Once the designer is satisfied with the *User Design Profile*, the *suitability portion* of the Selection process is executed. The design profile is compared to the data base of BIT functions to generate a list of BIT techniques suitable to the design or partition.

The techniques are then ranked through an estimation process. Estimated penalties for each suitable technique are calculated and an Estimation Penalty Report and Ranking is provided. This report shows, for each suitable technique, the estimated associated BIT penalty for each penalty category and for each technique. The overall ranking is computed using normalized penalty values followed by the application of the designer's weighting factors.

From the penalty report the designer chooses either the optimum technique or another technique of his/her own choice. If desired, the whole Selection process can be repeated using different parameter values to obtain a different ranking of penalties. The designer utilizes the functions in the CAD menu to insert the BIT circuit elements for the selected technique into the functional design schematic. The CAD menu functions aid the designer to: input data used to calculate the number of each BIT component required; provide a check-off diagram to keep track of those circuit components inserted; provide tutorial information to aid the interconnection process; generate the CAD component insertion command; add attributes onto the BIT circuit elements; and provide an estimate of the BIT penalties for each BIT group. The circuitry inserted for any partition is referred to as a BIT group.

An evaluation is performed after the insertion of the BIT circuitry. It provides a report of the actual BIT penalties (power, area, etc.) incurred due to the insertion of the BIT circuitry. If multiple partitions have been used, each circuit group and its penalties will be shown. A summary of the BIT penalties (added power, weight, etc. due to the BIT circuit elements) is also provided.

CAD-BIT provides an estimation of BIT penalties for each suitable technique. This estimation precedes the BIT selection and aids the technique selection process. The evaluation consists of the actual BIT penalties after the BIT circuitry has been inserted into the design.

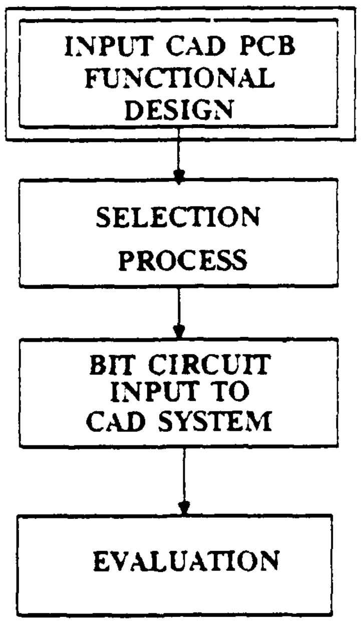

INPUT CAD PCB FUNCTIONAL DESIGN
SELECTION PROCESS
BIT CIRCUIT INPUT TO CAD SYSTEM
EVALUATION

Figure 1-1
CAD-BIT PROCEDURE OVERVIEW

1.2.2 CAD-BIT STRUCTURED OVERVIEW

The CAD-BIT procedure is divided into two environments, the operating system environment and the host CAD system environment, Figure 1-2. There are eight functions in the operating system environment and five in the host CAD system environment. The designer moves from one environment to the other as required by the CAD-BIT procedure.

If the designer wishes to obtain a brief description of CAD-BIT, Function 1 (Overview) is chosen at the operating system level. This will print the descrption on the screen.

Functions 2 and 3 of the operating system environment provide tutorials on the various BIT techniques in the CAD-BIT data base. Function 2 contains short tutorials for each technique as well as figures which can be viewed through Function 2 or Function 3 in the host CAD system environment. Function 3 of the operating system environment provides a more lengthy set of tutorials on the various techniques. There are seven long tutorials per technique. These include : advantages, disadvantages, bit sequences, bibliographies, parts data, attributes, and default design. As with the short tutorials, there are also figures available in the host CAD system environment.

The Selection process, as described in section 3.4.5, occurs as Function 4 in the operating system environment. Function 5 (BIT Insertion) is then invoked once the designer is satisfied with the BIT technique which has been selected. This operating system function generates a host CAD system command, accessed using CAD Function 3, to insert a calculated quantity of BIT circuit elements into the functional design schematic.

Once all the BIT circuit elements have been inserted, a host CAD system data extraction procedure is performed to obtain information from the functional design schematic. This procedure is accessed through Function 4 of the host CAD system environment. The output is fed into Function 6 of the operating system environment. This function generates the report of actual BIT penalties. If the designer is satisfied with the results, CAD-BIT can then be exited through Function 5 of the host CAD system and Function 8 of the operating system.

If alterations or additions need to be made to the CAD-BIT data base, utilities to accomplish these changes are provided through Function 7 of the operating system environment. Changes may include the addition of new BIT techniques, alteration of existing BIT techniques, etc. Section 3.4.8 contains descriptions of such utilities.

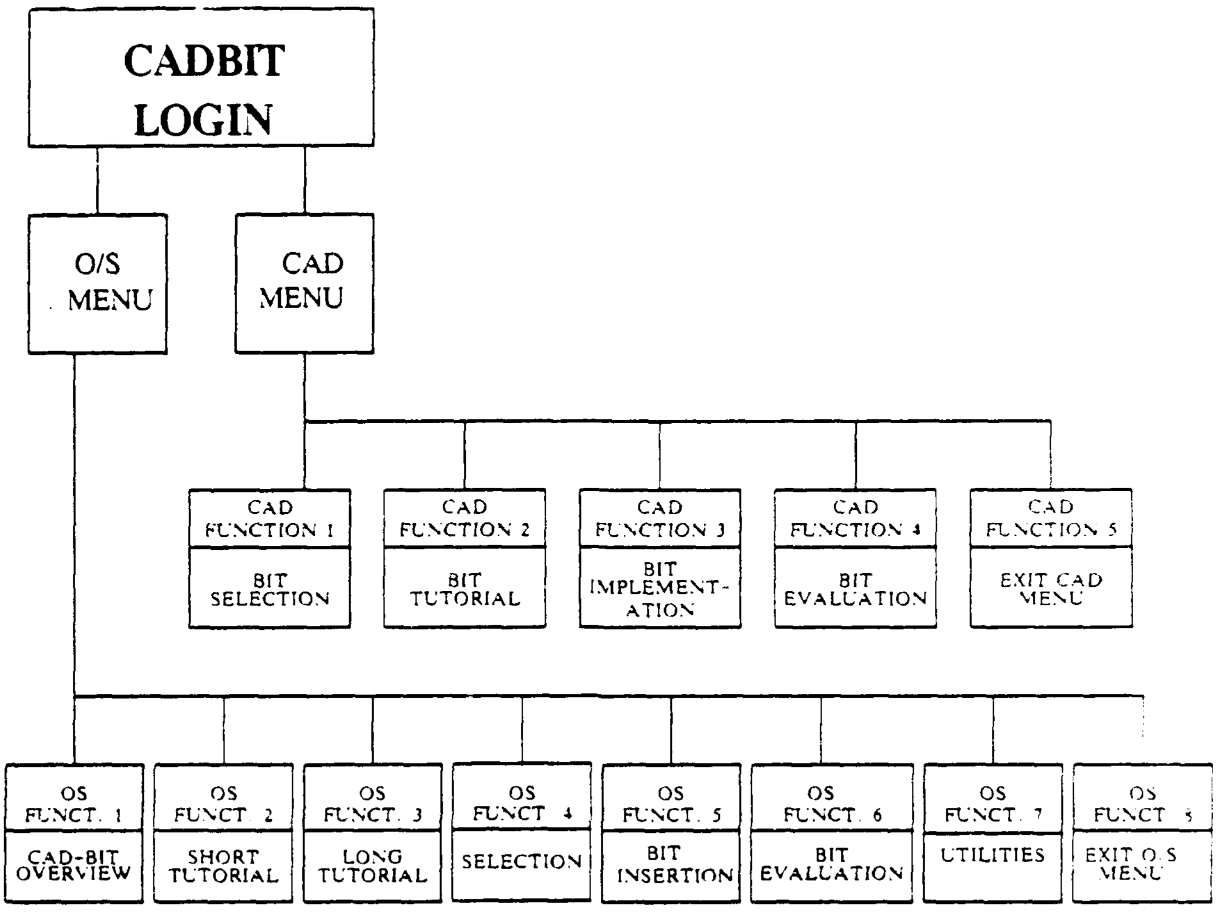


Figure 1–2

CAD–BIT STRUCTURE

2.0 APPLICABLE DOCUMENTS

The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superceding requirement.

2.1 GOVERNMENT DOCUMENTS

Military Standards

DOD-STD-2167A	Defense System
29 February 1988	Software Development

2.2 NON-GOVERNMENT DOCUMENTS

Final Report, Volume I Technical Issues

CAD-BIT Final Report	Grumman Aircraft
29 July 1988	Systems Division

Final Report, Volume II BIT Library

CAD-BIT Final Report	Grumman Aircraft
29 July 1988	Systems Division

3.0 REQUIREMENTS

The following paragraphs and subparagraphs specify all the programming requirements to ensure proper development of the CAD-BIT system.

3.1 PROGRAMMING REQUIREMENTS

3.1.1 PROGRAMMING LANGUAGES

Two programming languages are used in the CAD-BIT system, C and UNIX C-Shell. The C-Shell programs are used as utilities for the creation or alteration of C programs used in the CAD-BIT system. The C programs make up half of the CAD-BIT system and are used in the Selection procedure, the Bit Insertion procedure, and the Evaluation procedure. It is also required that a data extract program is resident on the implemented host system.

3.1.1.1 C

UNIX has been selected as the operating system for CAD-BIT because of its high degree of portability. Since nearly all of UNIX is written in the C language, C is the most appropriate language for developing CAD-BIT. While it is a general purpose structured programming language, it was also designed with systems applications in mind and, as such, provides the user with an enormous amount of power and flexibility. Since extensibility is a key feature of CAD-BIT, the flexibility which C provides is another reason why C forms a major part of the CAD-BIT system.

3.1.1.2 C-Shell

The C-Shell is one of UNIX's most important components. A shell is an interface. It has no special status so it can be revised or replaced with relatively little trouble. As with the C language, extensibility is of utmost importance, so, for this reason the C-Shell is used to generate the CAD-BIT operating system (UNIX) menus as well as the utilities for altering the CAD-BIT data base (see section 3.4.8).

3.1.1.3 CAD System Data Extraction

The data extraction program(s) are used to extract BIT circuit design data for use by the Evaluation program. These programs are host CAD system dependent and are user developed. Unique data extraction languages, compilers, and/or subroutines providing access to the CAD data base are generally provided by the CAD vendor. The use of the extracted data is described later in section 3.4.7.1.

3.1.2 COMPILER/ASSEMBLER

A C compiler is required for the CAD-BIT system and is generally available on UNIX based systems. However, no compiler is required for the C-Shells. The CAD host system must also have available any other compilers necessary for system dependent tasks such as data extraction program(s).

3.1.3 PROGRAMMING STANDARDS

The programming standards for the CAD-BIT system shall be in accordance with Appendix B of DOD-STD-2167A.

3.2 DESIGN REQUIREMENTS

3.2.1 SIZING AND TIMING REQUIREMENTS

Table 3-1 lists the estimates for sizing and timing requirements allocated to each of the CAD-BIT operating system functions and subfunctions. In cases where operations include CAD system commands, timing is dependent upon the CAD system. Memory locations will be determined by the host operating system, loaders, etc.

3.2.2 DESIGN STANDARDS

The design standards below shall be applied to the CAD-BIT system :

- Ease of *changes and/or additions* (*extensibility*).
- Shared data placed in a common block so that only one change needs to be made if necessary.
- Uniformity of code.

3.2.3 DESIGN CONSTRAINTS

There are no design constraints imposed upon the CAD-BIT system except for items which will inhibit the use of generic CAD systems.

3.2.4 FILE NAMES

The file names shown in Table 3-2 are those used to identify the code which generates CAD-BIT. All file names are shown in italics, lowercase type. The use of "*<variable>*" denotes a substitution of *variable* with a specific name. When a ".c" follows a file name, then that file is C source code. All other files are either data files, executeable code (CAD-BIT commands), or C-Shell scripts. Any UNIX commands used are represented in this document in bold, lowercase type.

Table 3-1

Sizing & Timing Requirements

(Estimates for O/S Functions only)

Functions & Subfunctions		Timing (in seconds)	Sizing (in kilobytes)	
			Software	Data Files
Main Menu		–	32	15
F1	Overview	1/2	–	1
F2	Short Tutorials	1/2	–	1*
F3	Long Tutorials	1/2	–	6*
F4	Selection	–	–	–
	Profile	2	32	1
	Suitability	1	32	14
	Penalty	2	64	2
F5	Insertion	1	32	4
F6	Evaluation	1	32	2
F7	Utilities	3	32**	–

* = per technique

** = per utility

Table 3–2
File Names / Functions

File Name(s)	Functions / Subfunctions
.cadbit	Selection/Profile
<cadsys>/cadbit/<t##>/<fign>	Short/Long Tutorial Figure Files
data/cadbit-eval-data	Evaluation
data/default-parts	Bit Insertion, Utilities
data/host-commands	Bit Insertion
data/penalty-cases	Selection/Penalty
data/ppd	Evaluation
data/technique-list	Main Menu, Selection, Bit Insertion
data/template	Selection/Profile, Utilities
data/<profilename>/btpr	Evaluation
data/<profilename>/penalty-report	Selection/Penalty
data/<profilename>/rejects	Selection/Suitability
data/<profilename>/suitable	Selection/Suitability
data/<profilename>–<t##>	Selection/Penalty, Bit Insertion
eval/cadbit-eval.c	Evaluation
help/<variable>–help	Selection/Profile
os-menu/cadbit	Main Menu
os-menu/displaymenu	Main Menu
os-menu/insert-bit	Bit Insertion
os-menu/option	Main Menu
os-menu/overview	Overview
profile/<profilename>/<profilename>.cut	Selection
profile/<profilename>/<t##>–urd	Selection/Penalty
selection/cadbit.c	Selection/Profile
selection/gen.c	Selection/Profile
selection/newbit.c	Selection/Profile
selection/override.c	Selection/Suitability
selection/penalty.c	Selection/Penalty
selection/suitable.c	Selection/Suitability
selection/syntax.c	Selection/Profile
selection/weighting-factors.c	Selection/Penalty
<t##>/circuitmenu	Bit Insertion, Utilities
<t##>/formulas.c	Selection/Penalty
<t##>/insert/<circuit>	Bit Insertion
<t##>/long-tut/<option>	Long Tutorials
<t##>/short-tut	Short Tutorials
<t##>/teq-file	Selection/Suitability, Utilities
utilities/add-technique	Utilities
utilities/fix-penalty	Utilities
utilities/gen-circuit	Utilities

3.3 INTERFACE IDENTIFICATION AND RELATIONSHIPS

There are 4 interfaces for the CAD-BIT system. Figure 3-1 is a pictorial representation of the interface relationships and Table 3-3 is a brief description. A more detailed description is found below :

3.3.1 USER / CAD-BIT OPERATING SYSTEM ENVIRONMENT

The User / Operating System environment consists of user input into the CAD-BIT operating system. The six areas listed illustrate this interface :

- CAD-BIT menu selections.
- Overview and Tutorial (text) generation.
- Selection procedure (profile generation, suitability determination, penalty calculations).
- BIT Insertion ("DO" command generation).
- BIT Evaluation report.
- Utilities (add-technique, gen-circuit, fix-penalty).

3.3.2 USER / CAD ENVIRONMENT

The User / CAD environment interface consists of user input of CAD commands into the host CAD graphics environment. The three areas listed illustrate this interface :

- Insertion of tutorial figures into the host CAD graphics environment.
- Insertion of BIT Technique Insertion Diagrams (BTIDs) into the host CAD graphics environment.
- Insertion of the PCB functional design into the host CAD graphics environment.

3.3.3 CAD-BIT OPERATING SYSTEM / CAD ENVIRONMENT

The CAD-BIT operating system environment / CAD environment consists of any interaction between CAD-BIT operating system functions and CAD functions. The two areas listed illustrate this interface :

- BIT Insertion into PCB via "DO" commands.
- Generation of BIT Technique Penalty Report (BTPR) of BIT data extracted from PCB via host CAD extraction program.

3.3.4 CAD-BIT OPERATING SYSTEM / UNIX OPERATING SYSTEM ENVIRONMENT

The CAD-BIT operating system / UNIX operating system environment consists of any interaction between CAD-BIT functions and UNIX functions. This interaction occurs when CAD-BIT uses a UNIX command in its function(s). One occurrence is in the use of UNIX's "more" command on the CAD-BIT Overview, Short Tutorials, and Long Tutorials. This UNIX command displays the contents of a specified data file a screenful at a time in the window from which the command was issued.

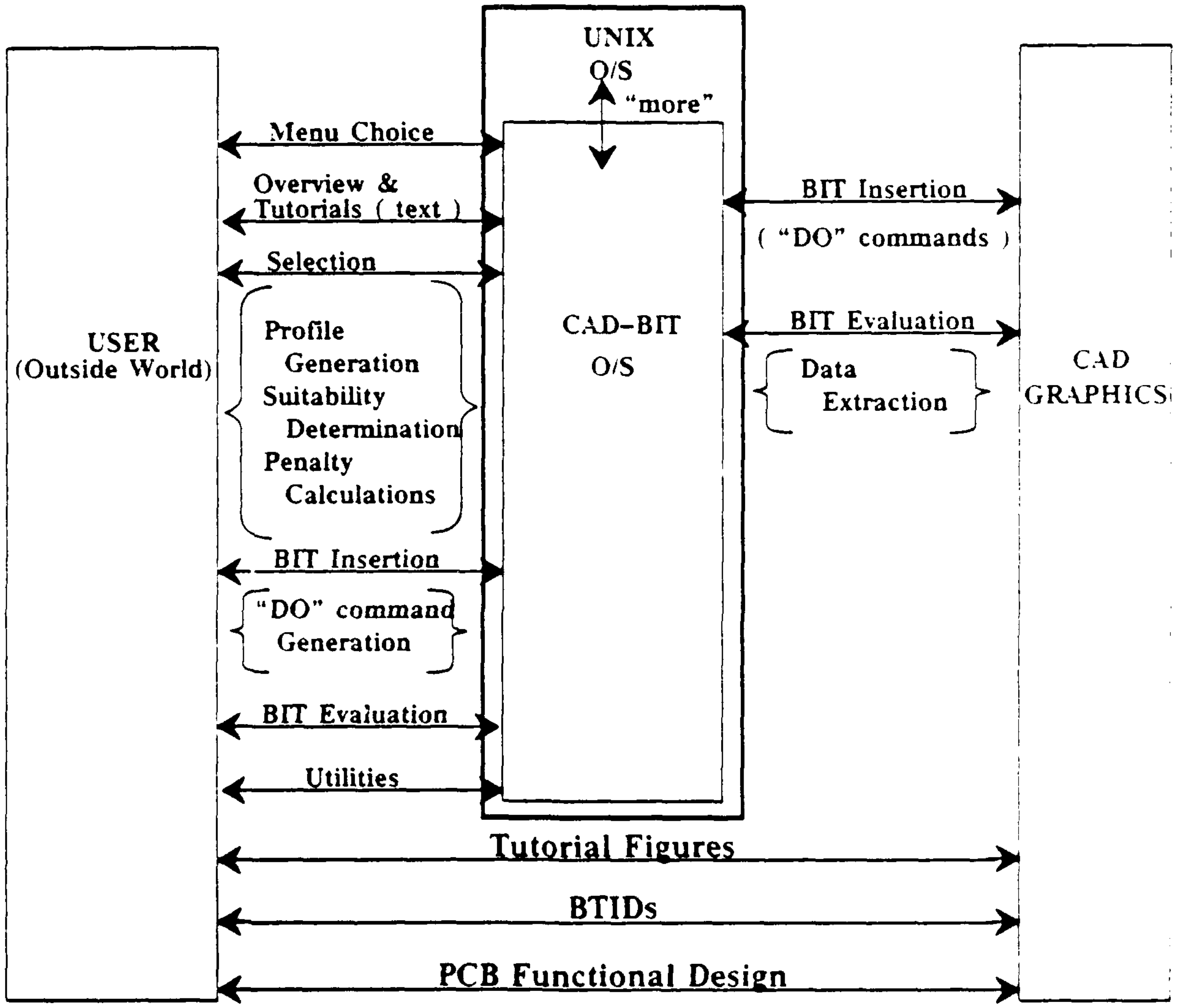


Figure 3-1

CAD-BIT INTERFACE DIAGRAM

Table 3-3
CAD-BIT Interface Relationships

Interface	Condition Present	Input	Output
User / CAD-BIT Operating System	Menu choice	User inputs menu choice.	Appropriate procedure accessed.
	Tutorials (text)	User inputs tutorial to be displayed.	Tutorial is displayed on operating system screen.
	Selection - Profile - Suitability - Penalty	User inputs PCB attributes and penalty parameters.	Profile displayed, Suitable BIT Techniques determined and displayed, and Penalty Report displayed.
	"DO" command Generation	User chooses the technique and circuit desired.	CAD host graphics execute file created containing insertion command.
	BIT Technique Penalty Report	User runs Evaluation C program on data extract file.	BIT Technique Penalty Report generated to operating system screen.
	Utilities	User inputs information into utility.	Program is generated which will be run in the operating system.
User / CAD	Tutorial Figures	User indicates figure and digitizes where it is to be placed in CAD host graphics environment.	Tutorial figure appears in CAD host graphics environment.
	BTID Insertion	User indicates BTID and digitizes where it is to be placed in CAD host graphics environment.	BTID appears in CAD host graphics environment
	PCB Functional Design	User inputs PCB functional design into CAD environment.	CAD file which is ready for CAD-BIT
CAD-BIT Operating System / CAD	BIT Insertion	"DO" command execute file run in CAD host graphics environment.	Appropriate BIT circuit(s) inserted into digitized place in PCB.
	BIT Evaluation	Data extract run on PCB in CAD host graphics environment.	Data file of BIT information generated to be used in BIT Technique Penalty Report
CAD-BIT Operating System / UNIX Operating System	Overview or Tutorial selected.	Data file to be printed to terminal display.	Screen by screen display of data file.

3.4 FUNCTIONAL AND PERFORMANCE REQUIREMENTS

This section defines the functional and performance requirements of the CAD-BIT system. Table 3-4 indicates the types of code used in each function and subfunction of the CAD-BIT system as well as the command(s) used to initiate them. In the following subparagraphs references will be made to inputs consisting of particular BIT Technique data. This data is found in Volume II, Data Base of BIT Functions, except for the samples that are included in this Volume. The operations performed on the data files are outlined as well as the outputs generated. Except where noted, all files are assumed to be located under a common CAD-BIT directory. Files and/or directory levels are italicized. Variable names are denoted by "< >". Refer back to Table 3-2 for clarification of any file names.

3.4.1 CAD-BIT O/S MENU

The CAD-BIT operating system menu is located in the host system's UNIX environment. All of CAD-BIT's UNIX driven functions are contained in this menu. The items are numbered in the order that they are to be performed. See Figures 3-2 and 3-3 for the setup of this menu.

(1) Inputs

Assuming the user is logged into the host system and has a PCB schematic active, the CAD-BIT operating system (UNIX) menu is activated by typing the following two commands after the system prompt :

- scripts
- cadbit

The command "scripts" is an alias initialized in the CAD system login files which changes the working directory to *os-menu*. The command "cadbit" executes the C-Shell script *os-menu/cadbit* which sets up the CAD-BIT main menus. See Figures 3-2 and 3-3 for a sample setup of the CAD-BIT operating system and CAD main menu screens.

(2) Processing

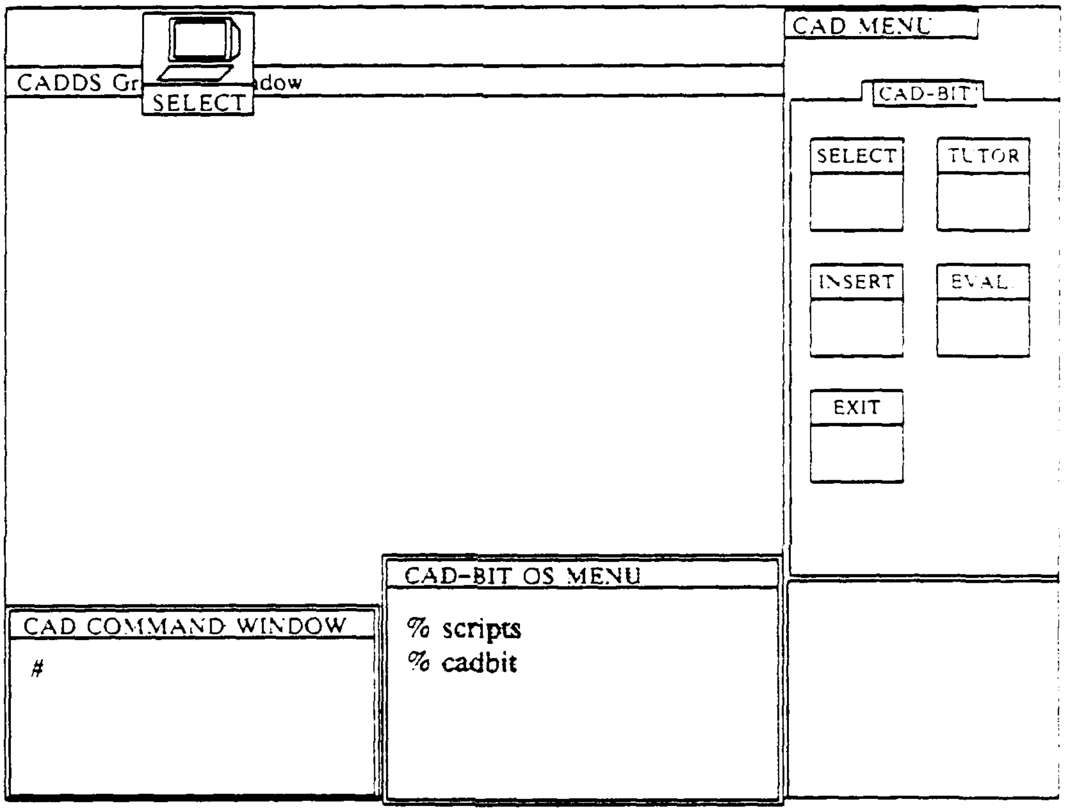
This routine prints out the "CAD-BIT O/S MENU" heading and then executes the C-Shell script *os-menu/displaymenu* which displays all of the menu's options. Once the CAD-BIT operating system (UNIX) menu is activated, the user makes subsequent menu selections by typing in numbers corresponding to the menu item being selected. The choice is read into the C-Shell script *os-menu/option* which executes the programs and/or displays data files corresponding to the menu choice.

(3) Outputs

The CAD-BIT procedures chosen by the user are generated as a result of the CAD-BIT O/S menu. Each of the CAD-BIT operating system's menu items are explained in the following sections. The main menu items correspond to the O/S menu functions in Figure 1-2.

Table 3-4 : Software Module Summary

Functions & Subfunctions		Command(s)	Code
	Main Menu	*cadbit* / *displaymenu* / *option*	C-Shell
F1	Overview	—	—
F2	Short Tutorials	*short-tut*	C-Shell
F3	Long Tutorials	*long-tut*	C-Shell
F4	Selection	—	—
	Profile	*cadbit* / *newbit* / *gen*	C
	Suitability	*suitable* / *override*	C
	Penalty	*penalty* / *weights* / *<t##>_form*	C
F5	Insertion	*search*	C
		<t##>/insert/<circuit>	host CAD system
F6	Evaluation	*cadbit-eval*	C
F7	Utilities	*add-technique* / *gen-circuit* / *fix-penalty*	C-Shell



```
% scripts
% cadbit
```

Figure 3-2
CAD-BIT MODULE OPERATING SYSTEM MENU SETUP

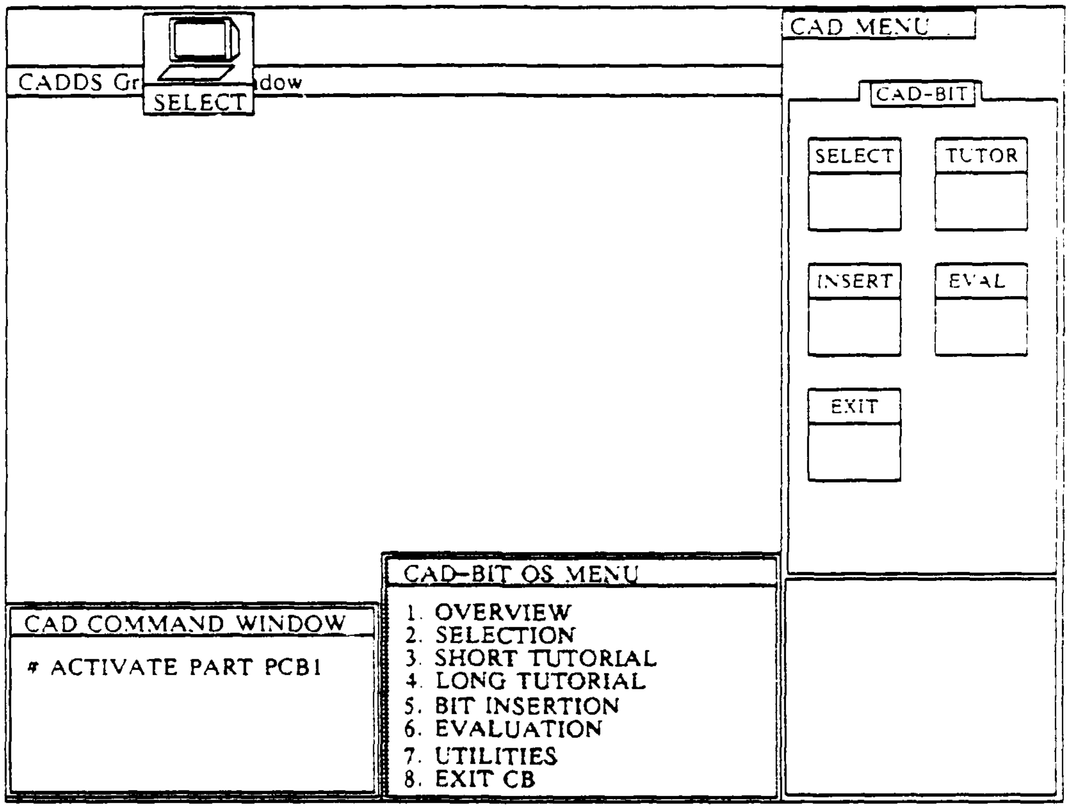


Figure 3–3
CAD–BIT MODULE OPERATING SYSTEM MAIN MENUS

3.4.2 CAD-BIT OVERVIEW

The CAD-BIT Overview is a menu item chosen from the main CAD-BIT operating system (UNIX) menu. The CAD-BIT Overview provides a brief description of the function of the CAD-BIT system.

(1) Inputs

The input data for the CAD-BIT Overview is a standard UNIX text file containing the information to be displayed. The input file name is *os-menu/overview*.

(2) Processing

The processing of the CAD-BIT Overview consists of the calling of the standard UNIX command "**more**" from the main OS CAD-BIT menu (Function 1 of the O/S menu). See Figures 3-4 and 3-5 for a sample setup of the CAD-BIT Overview.

(3) Outputs

The output of the CAD-BIT Overview consists of the display of the overview description file on the display terminal. On a window system, the display will occur within the MAIN CAD-BIT Window (MCBW).

3.4.3 SHORT TUTORIAL

The CAD-BIT Short Tutorial is a menu item chosen from the main operating system (UNIX) menu. The CAD-BIT Short Tutorial provides a brief description of each of the BIT Techniques residing on the CAD-BIT system. The tutorial can contain both text and figures.

(1) Inputs

The input data for the CAD-BIT Short Tutorial is a standard UNIX text file containing the tutorial information to be displayed. The input file name is *<t##>/short-tut*, where *<t##>* represents the BIT Technique number. The figure files are contained in the directory <cadsys>*/cadbit/<t##>*, where *cadsys* is the host system parts directory, and in files */fig1*, */fig2*, ... */fign*. These will reside in the host CAD system's graphics file format. The Short Tutorial text and figures are included in Volume II, Data Base of BIT Functions.

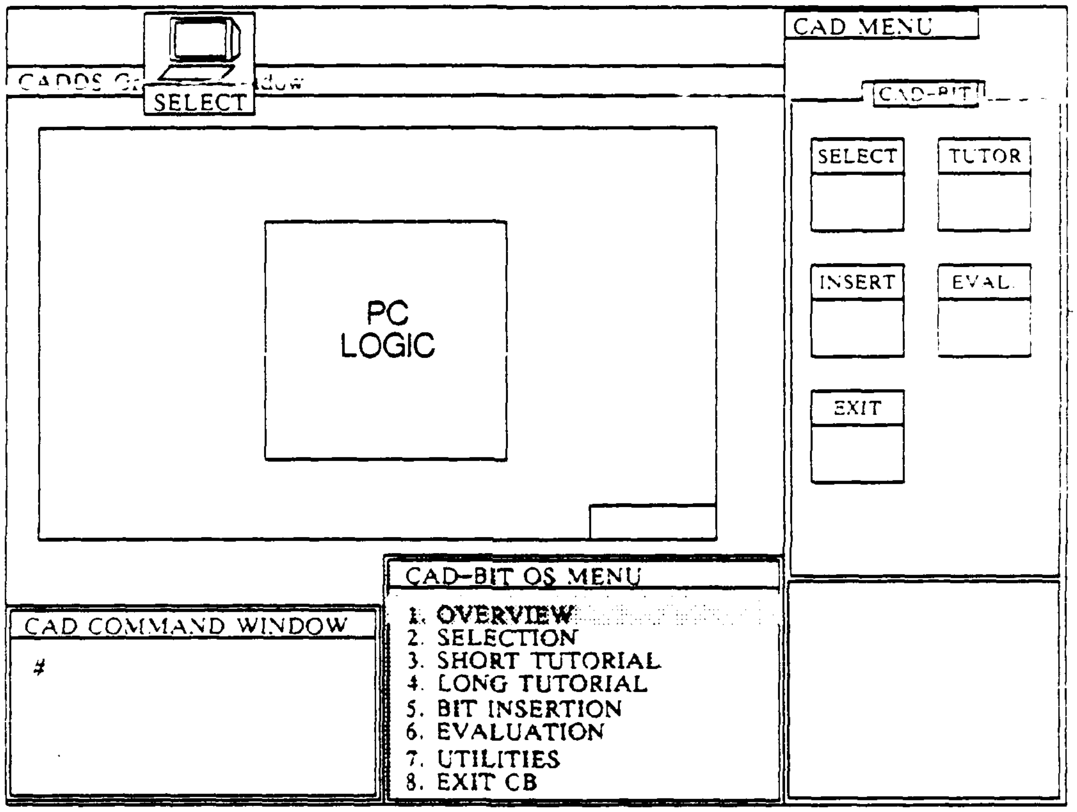
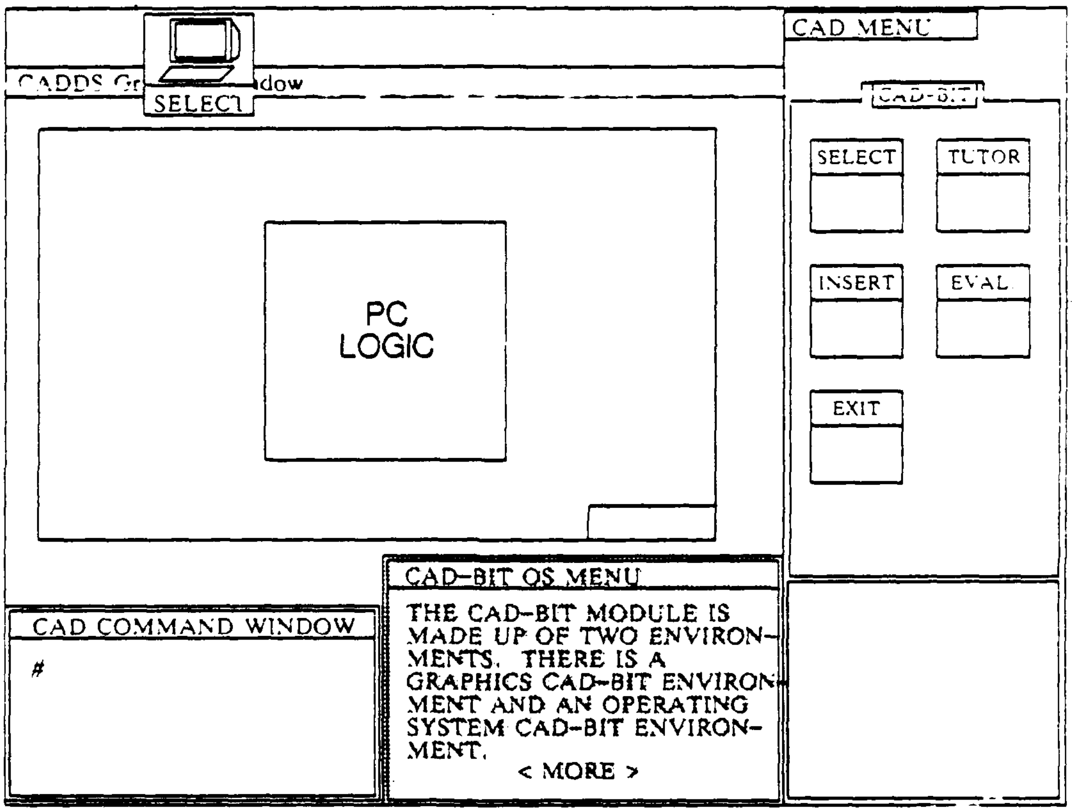


Figure 3–4
CAD–BIT OPERATING MODE: SELECTING OPERATING SYSTEM ENVIRONMENT OVERVIEW



THE CAD-BIT MODULE IS MADE UP OF TWO ENVIRONMENTS. THERE IS A GRAPHICS CAD-BIT ENVIRONMENT AND AN OPERATING SYSTEM CAD-BIT ENVIRONMENT.

< MORE >

Figure 3-5
OVERVIEW DISPLAY IN
OPERATING SYSTEM ENVIRONMENT

(2) Processing

The processing of the text portion of the CAD-BIT Short Tutorial, Figure 3-6, consists of the C-Shell script *os-menu/short-tut* issuing the standard UNIX command "**more**" from the main OS CAD-BIT menu (Function 2 of the O/S menu). The tutorial is identified by selecting the technique number of the technique to be displayed. These choices are contained in the file *data/technique-list* which is printed to the screen. When figures are referenced, they may be displayed by entering the graphics mode and inserting the figure file into the graphics display using a standard CAD system command. See Figures 3-7 through 3-13 for sample Short Tutorial selection screens.

(3) Outputs

The output of the CAD-BIT Short Tutorial consists of the display of the file *<t##>/short-tut* on the display terminal. On a window system, the display will occur within the MAIN CAD-BIT Window (MCBW).

3.4.4 LONG TUTORIAL

The CAD-BIT Long Tutorial is a menu item chosen from the main operating system (UNIX) menu. The CAD-BIT Long Tutorial provides a detailed description of each of the BIT Techniques residing on the CAD-BIT system. The tutorial can contain both text and figures.

(1) Inputs

The input data for the CAD-BIT Long Tutorial is a set of standard UNIX text files containing the tutorial information to be displayed. The input file names are *<t##>/long-tut/option*, where *<option>* can represent advantages, disadvantages, bit-sequence, attributes, default-design, parts-data, and/or bibliography. The figure files are contained in the directory *<cadsys>/cadbit/<t##>* and in files */fig1*, */fig2*, ... */fign*. They are referenced in various long tutorial text files. The Long Tutorial text and figures are included in Volume II, Data Base of BIT Functions.

(2) Processing

The processing of the text portion of the CAD-BIT Long Tutorial, Figure 3-14, consists of the C-Shell script *os-menu/long-tut* issuing the standard UNIX command "**more**" from the main OS CAD-BIT menu (Function 3 of the O/S menu). The tutorial is identified by selecting the technique number of the technique to be displayed. These choices are contained in the file *data/technique-list* which is printed to the screen. The user then selects the type of Long Tutorial desired. When figures are referenced, they may be displayed by

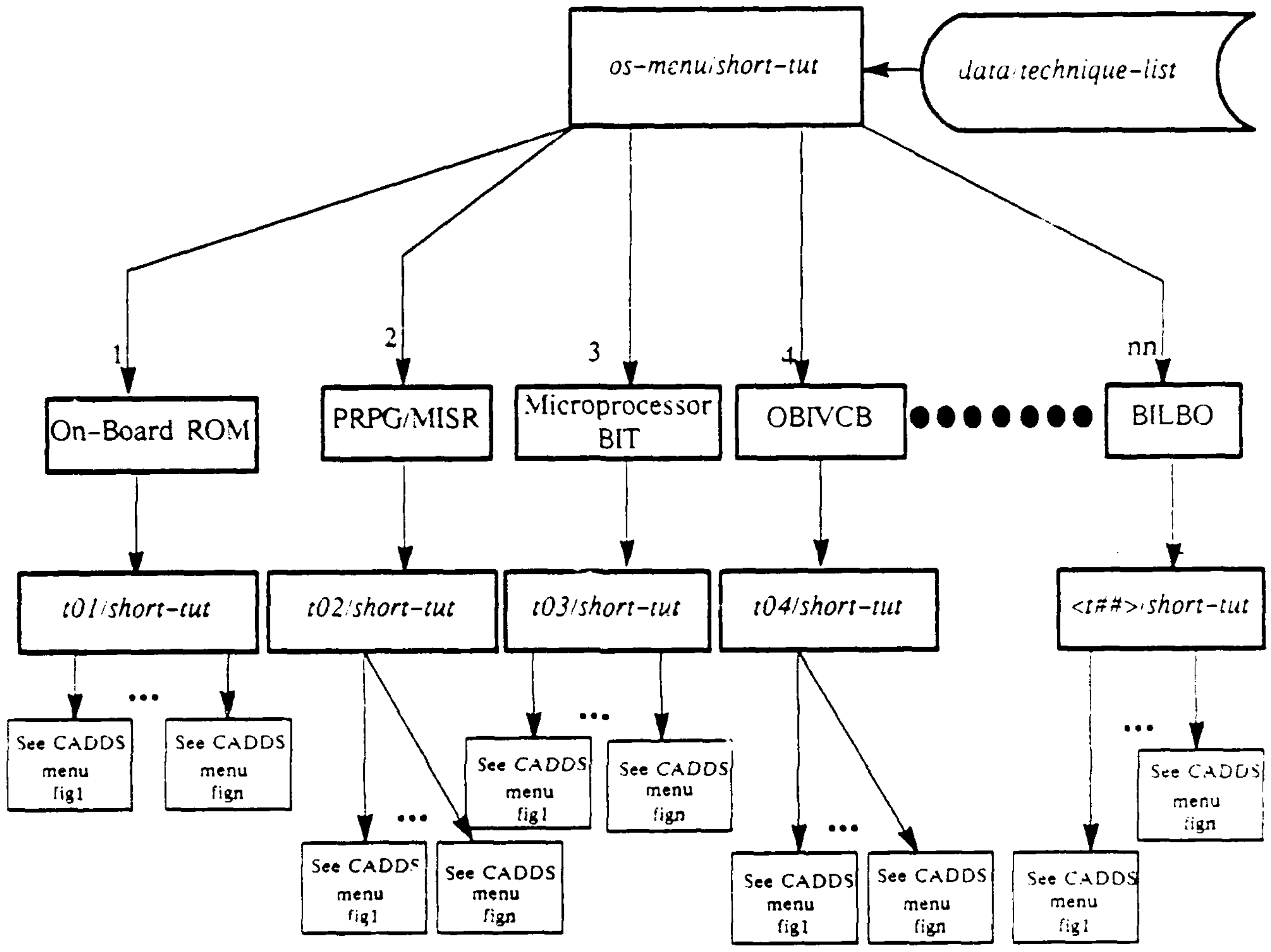


Figure 3–6
SHORT TUTORIAL FLOW DIAGRAM

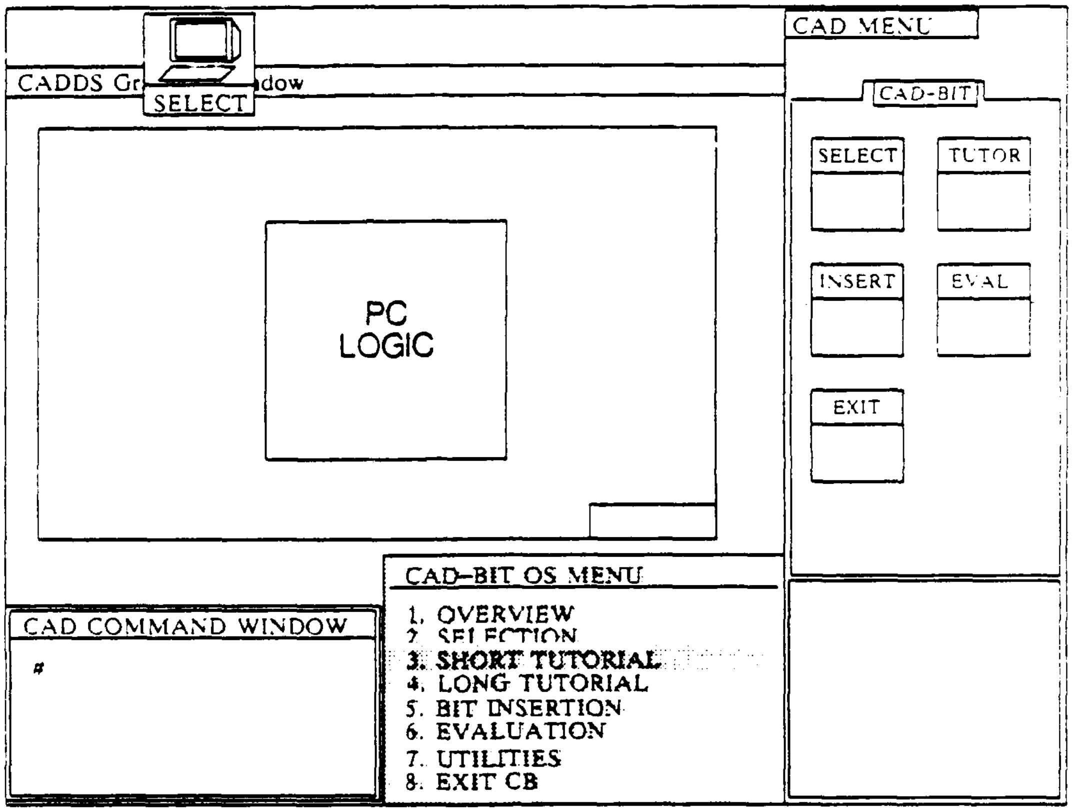
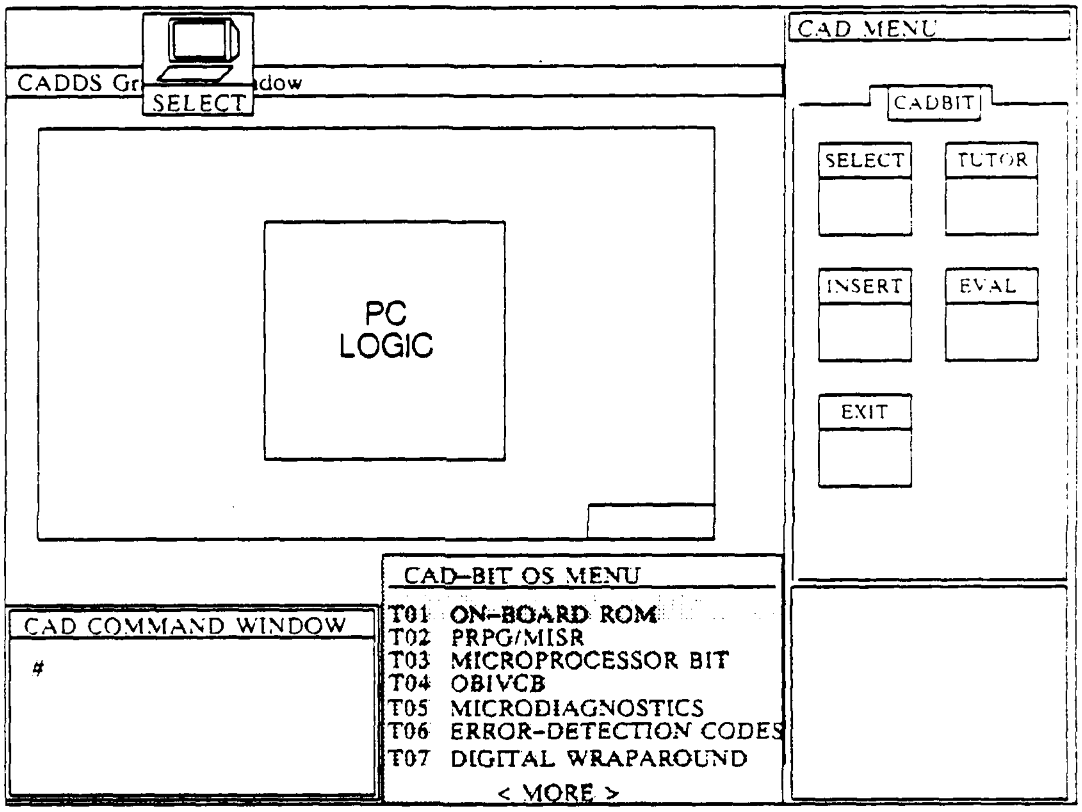
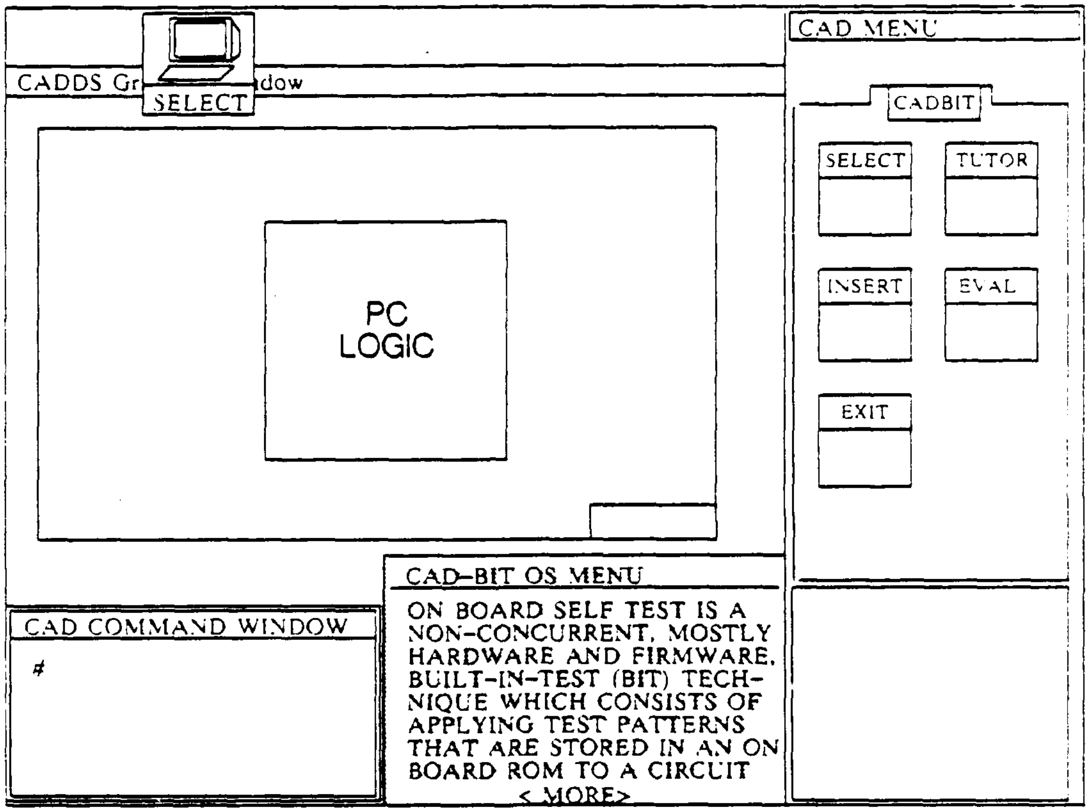


Figure 3–7
CAD–BIT OPERATING MODE: SELECTING OPERATING SYSTEM ENVIRONMENT SHORT TUTORIAL



T01 ON-BOARD ROM
T02 PRPG/MISR
T03 MICROPROCESSOR BIT
T04 OBIVCB
T05 MICRODIAGNOSTICS
T06 ERROR-DETECTION CODES
T07 DIGITAL WRAPAROUND
< MORE >

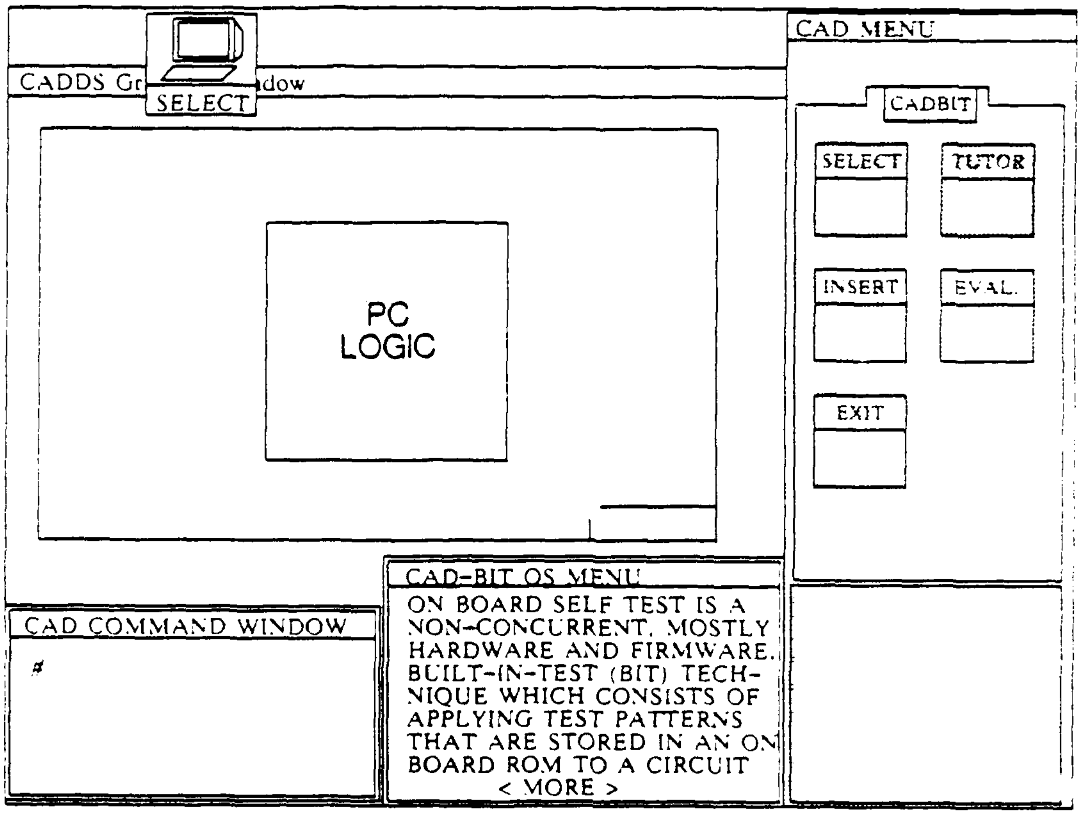
Figure 3-8
SHORT TUTORIAL TECHNIQUE MENU: SELECTING SHORT TUTORIAL TECHNIQUE



ON BOARD SELF TEST IS A NON-CONCURRENT, MOSTLY HARDWARE AND FIRMWARE, BUILT-IN-TEST (BIT) TECHNIQUE WHICH CONSISTS OF APPLYING TEST PATTERNS THAT ARE STORED IN AN ON BOARD ROM TO A CIRCUIT

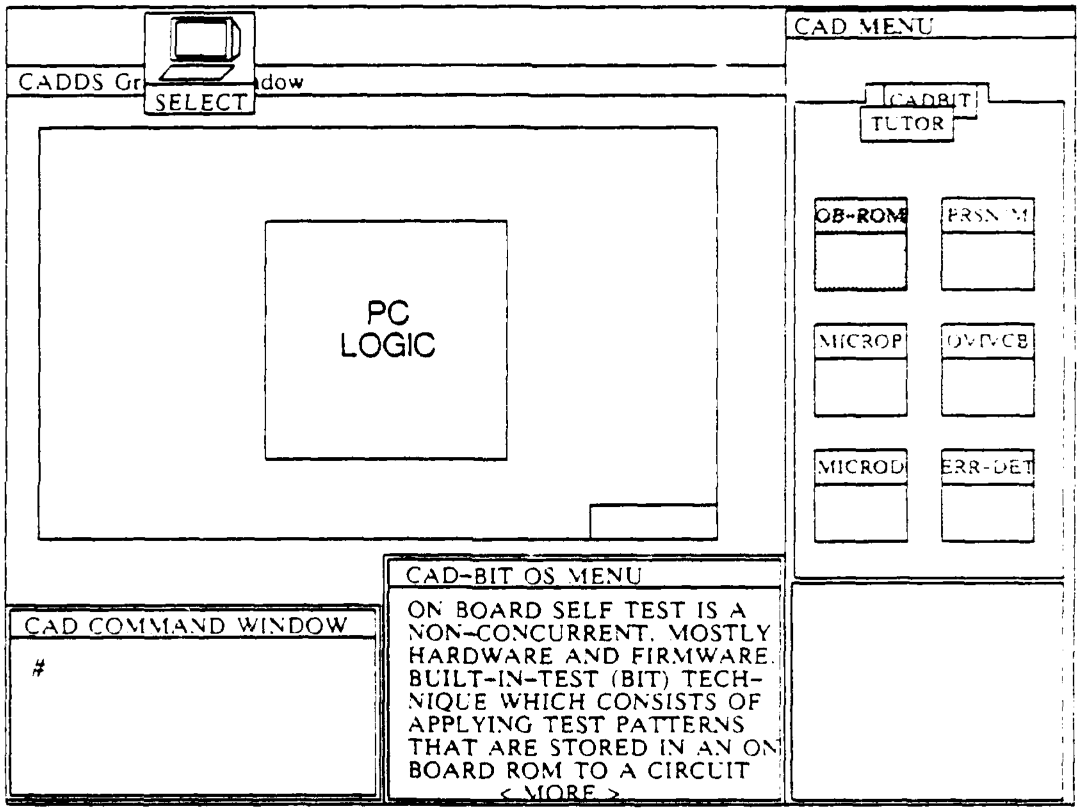
< MORE>

Figure 3-9
SHORT TUTORIAL TECHNIQUE DISPLAY IN OPERATING SYSTEM ENVIRONMENT



ON BOARD SELF TEST IS A NON-CONCURRENT, MOSTLY HARDWARE AND FIRMWARE, BUILT-IN-TEST (BIT) TECHNIQUE WHICH CONSISTS OF APPLYING TEST PATTERNS THAT ARE STORED IN AN ON BOARD ROM TO A CIRCUIT
< MORE >

Figure 3-10
CAD-BIT MENU TUTORIAL SELECTION IN CAD ENVIRONMENT

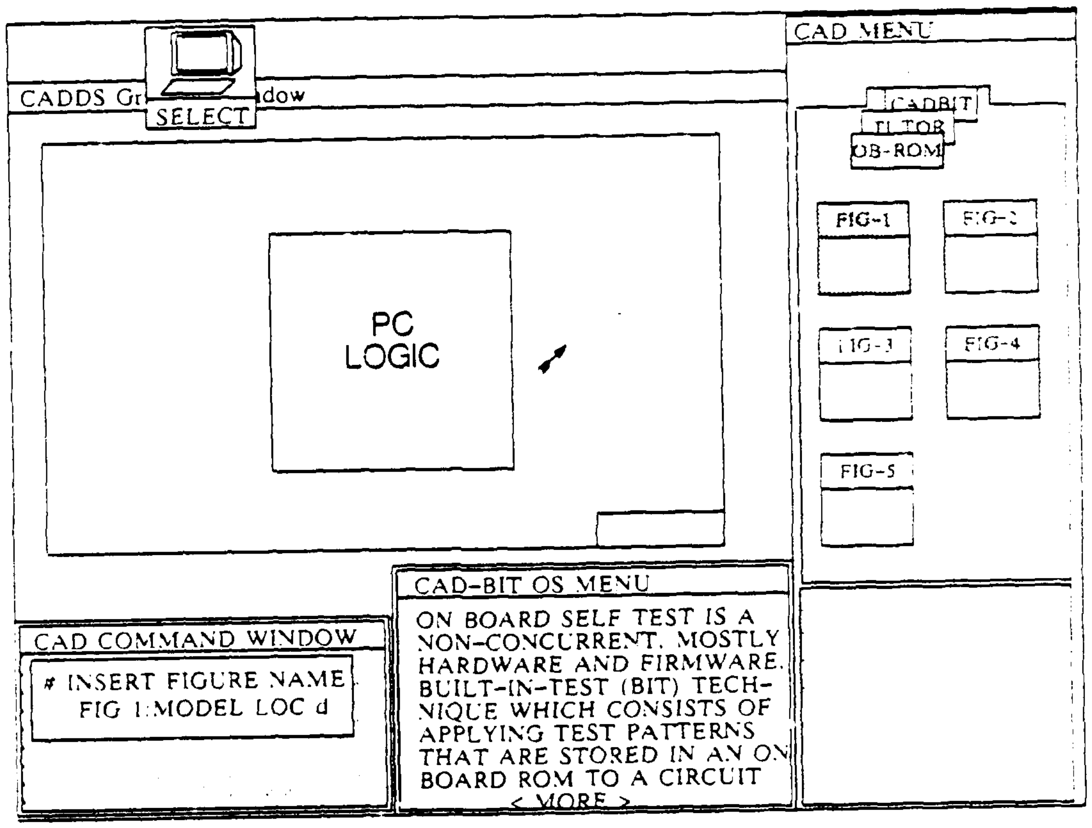


ON BOARD SELF TEST IS A NON-CONCURRENT. MOSTLY HARDWARE AND FIRMWARE. BUILT-IN-TEST (BIT) TECHNIQUE WHICH CONSISTS OF APPLYING TEST PATTERNS THAT ARE STORED IN AN ON BOARD ROM TO A CIRCUIT

< MORE >

Figure 3-11

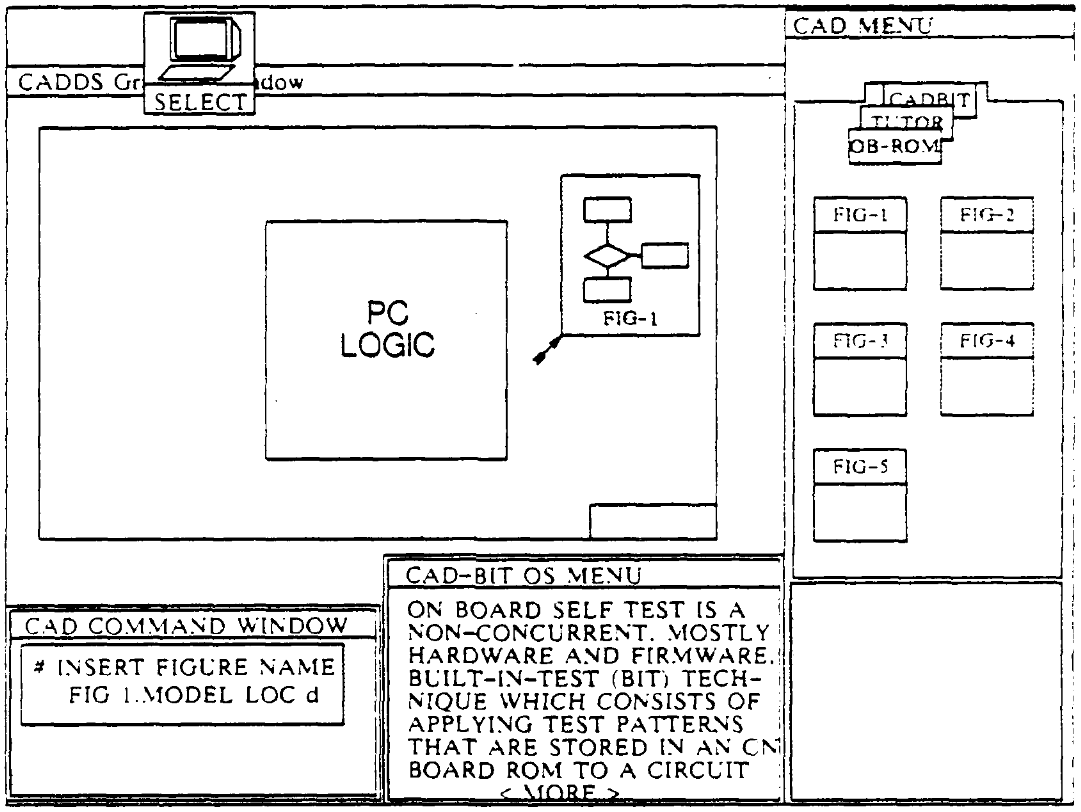
TUTORIAL TECHNIQUE SELECTION IN CAD ENVIRONMENT



INSERT FIGURE NAME
FIG 1:MODEL LOC d

ON BOARD SELF TEST IS A NON-CONCURRENT, MOSTLY HARDWARE AND FIRMWARE, BUILT-IN-TEST (BIT) TECHNIQUE WHICH CONSISTS OF APPLYING TEST PATTERNS THAT ARE STORED IN AN ON BOARD ROM TO A CIRCUIT

Figure 3–12
TUTORIAL FIGURE SELECTION IN CAD ENVIRONMENT



INSERT FIGURE NAME
FIG 1.MODEL LOC d

ON BOARD SELF TEST IS A NON-CONCURRENT, MOSTLY HARDWARE AND FIRMWARE, BUILT-IN-TEST (BIT) TECHNIQUE WHICH CONSISTS OF APPLYING TEST PATTERNS THAT ARE STORED IN AN CN BOARD ROM TO A CIRCUIT
< MORE >

Figure 3-13

TUTORIAL FIGURE PLACEMENT IN CAD ENVIRONMENT

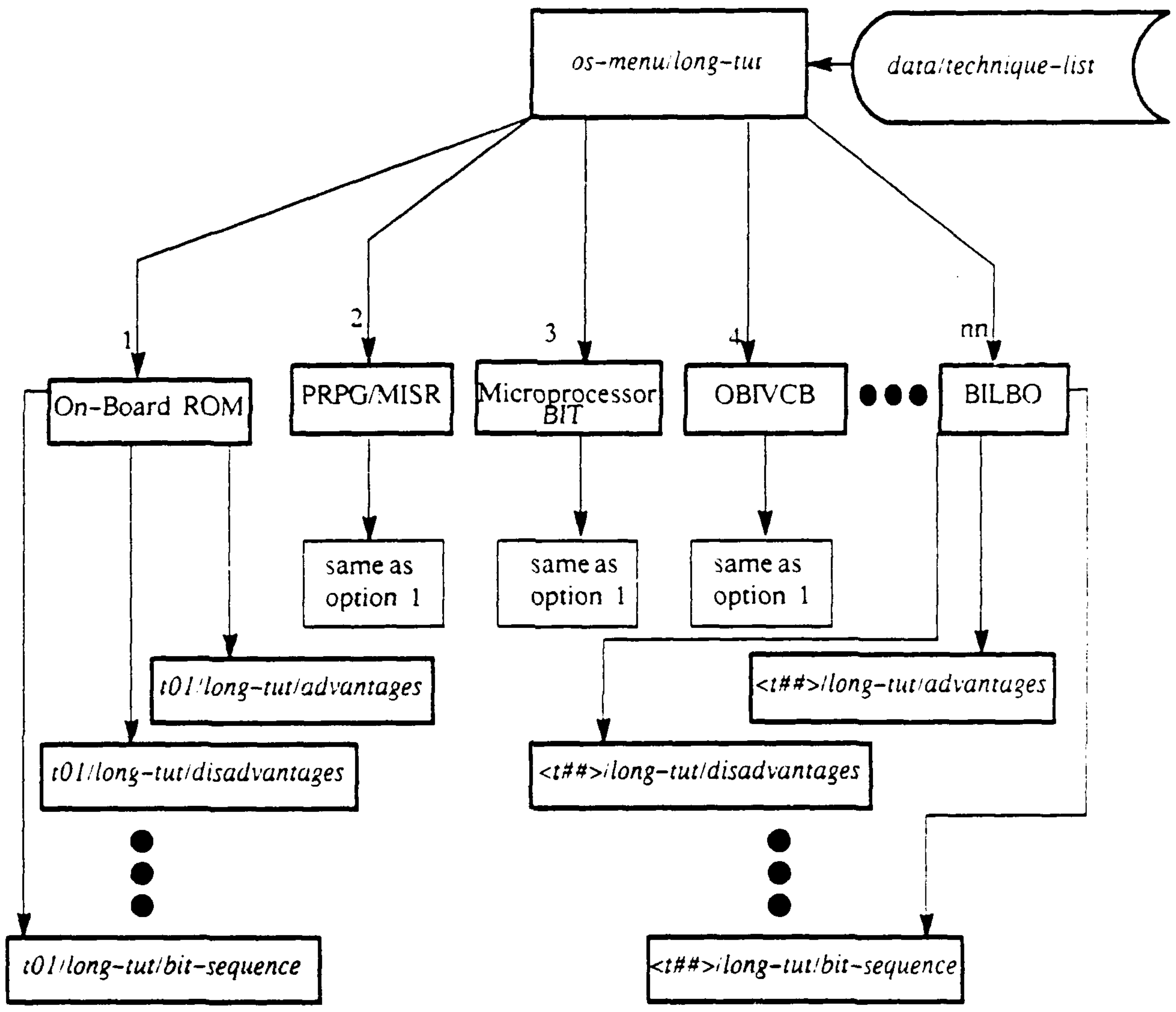


Figure 3-14

LONG TUTORIAL FLOW DIAGRAM

entering the graphics mode and inserting the figure file into the graphics display using a standard CAD system command. See Figures 3-15 through 3-18 for sample Long Tutorial selection screens.

(3) Outputs

The output of the CAD-BIT Long Tutorial consists of the display of the file(s) *<t##>/long-tut/<option>*. On a window system, the display will occur within the MAIN CAD-BIT Window (MCBW). There shall also be an option to send a hardcopy of the displayed file to the line printer.

3.4.5 SELECTION

Selection appears as Function 4 of the main operating system (UNIX) menu. In choosing this option the user is referred to the Selection Window for processing. Figures 3-19 through 3-20 illustrate the setup of the Selection Window. This procedure, from user input (User Design Profile) and calculations, selects suitable techniques, calculates BIT penalties, and generates a penalty report of all BIT techniques considered to be suitable for a particular PCB design. These techniques are ranked in decreasing order of preference. The user may follow the recommended technique or make another choice. This procedure is divided into three sections (Profile Generation, Suitability Determination, and Technique Ranking).

3.4.5.1 Profile Generation

This section generates a design profile of BIT technique attributes and penalty parameters for a particular PCB design.

(1) Inputs

The input data for the Profile generation section of the Selection procedure is the name of the profile to be generated and/or altered. This file is located in the directory *profile/<profilename>* and has the name */<profilename>.cat*. The data file *data/template* is also input into this procedure. This file contains the attribute/parameter variables, the default answers, the allowable answers, and the questions to prompt the user for the generation of the profile.

(2) Processing

The C program *selection/cadbit.c*, Figure 3-21, prompts the user for the name of the profile to be used. It then checks for the existence of this file under the appropriate direc-

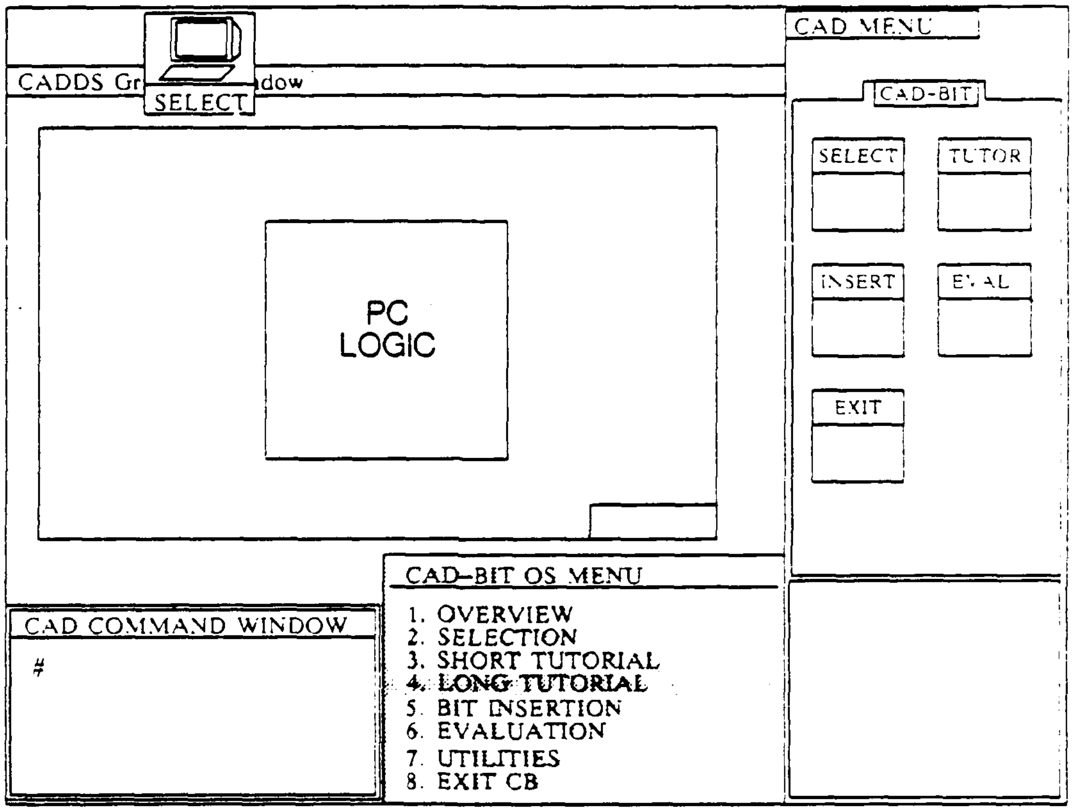
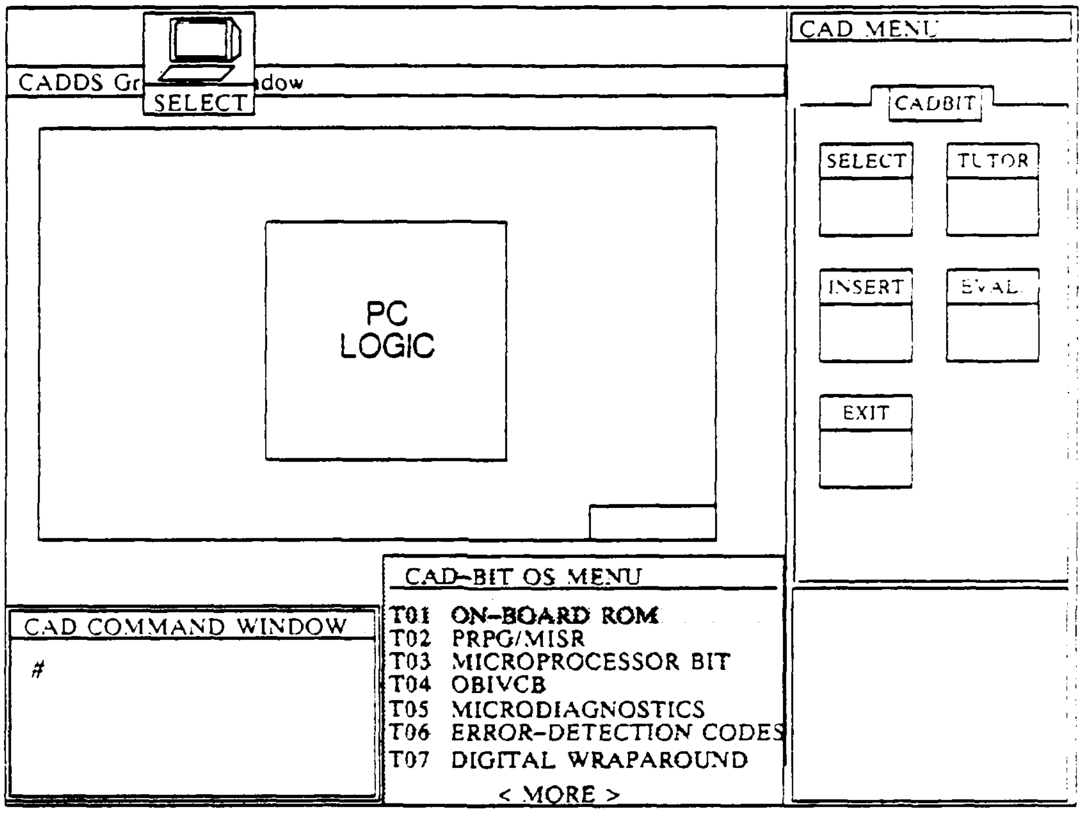


Figure 3-15

CAD-BIT OPERATING MODE: SELECTING OPERATING SYSTEM ENVIRONMENT LONG TUTORIAL



CAD-BIT OS MENU

```
T01  ON-BOARD ROM
T02  PRPG/MISR
T03  MICROPROCESSOR BIT
T04  OBIVCB
T05  MICRODIAGNOSTICS
T06  ERROR-DETECTION CODES
T07  DIGITAL WRAPAROUND
            < MORE >
```

Figure 3-16
LONG TUTORIAL TECHNIQUE MENU: SELECTING LONG TUTORIAL TECHNIQUE

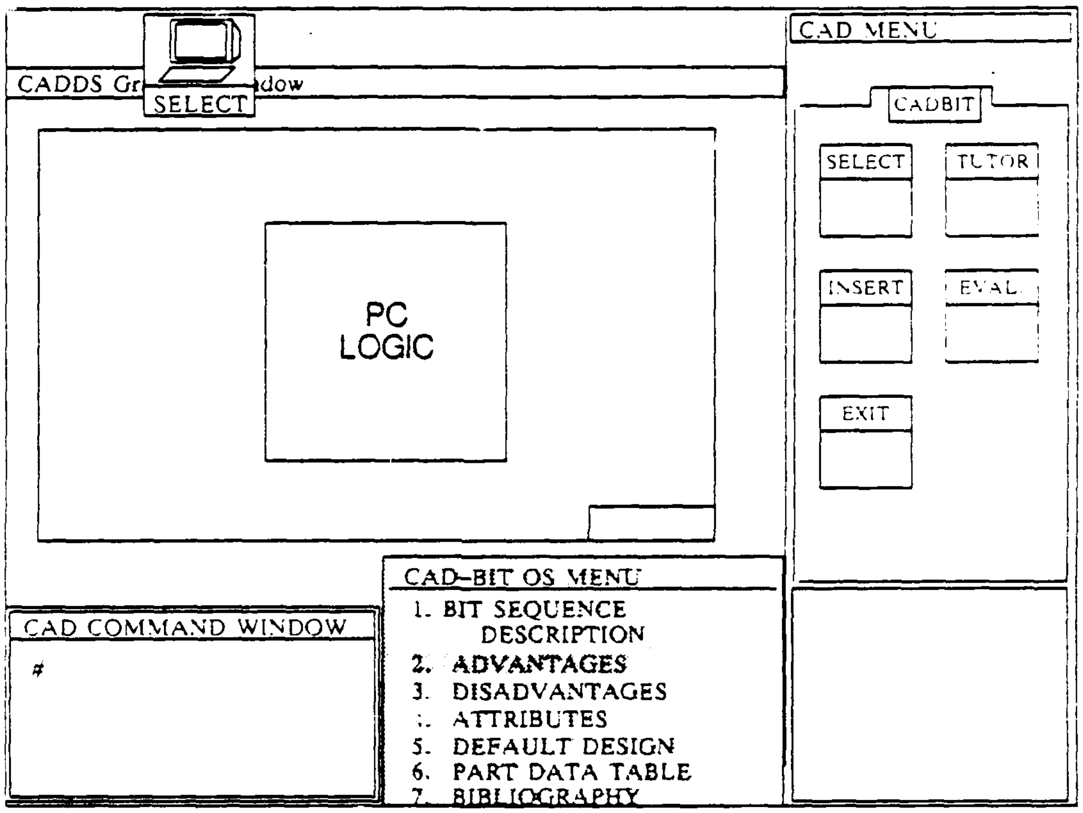
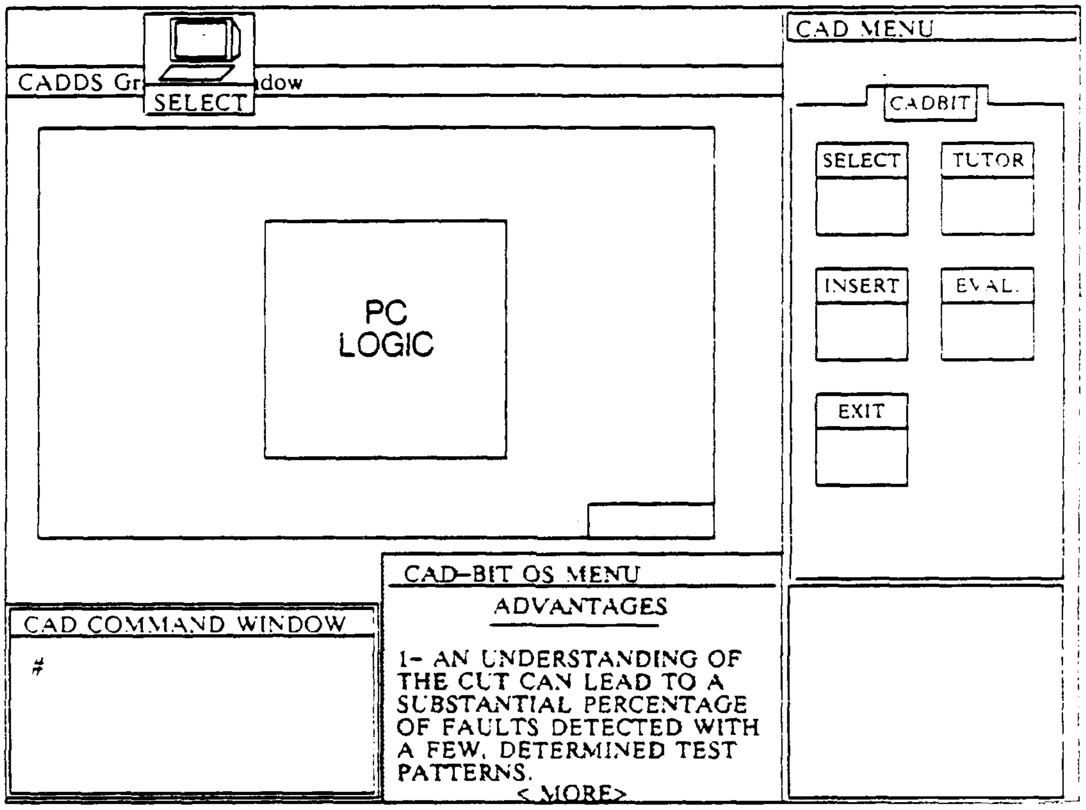


Figure 3-17
LONG TUTORIAL OPTIONS LIST



ADVANTAGES

1- AN UNDERSTANDING OF THE CUT CAN LEAD TO A SUBSTANTIAL PERCENTAGE OF FAULTS DETECTED WITH A FEW, DETERMINED TEST PATTERNS.

< MORE>

Figure 3-18

LONG TUTORIAL ADVANTAGES OPTION DISPLAY IN OPERATING SYSTEM ENVIRONMENT

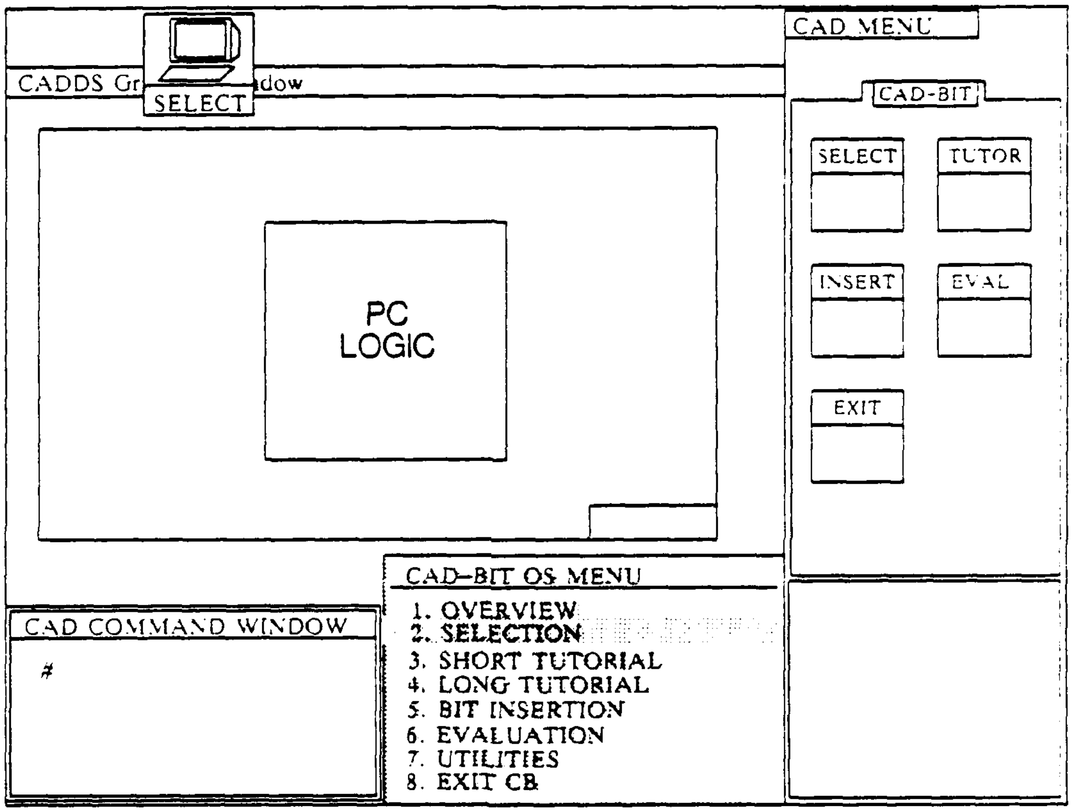
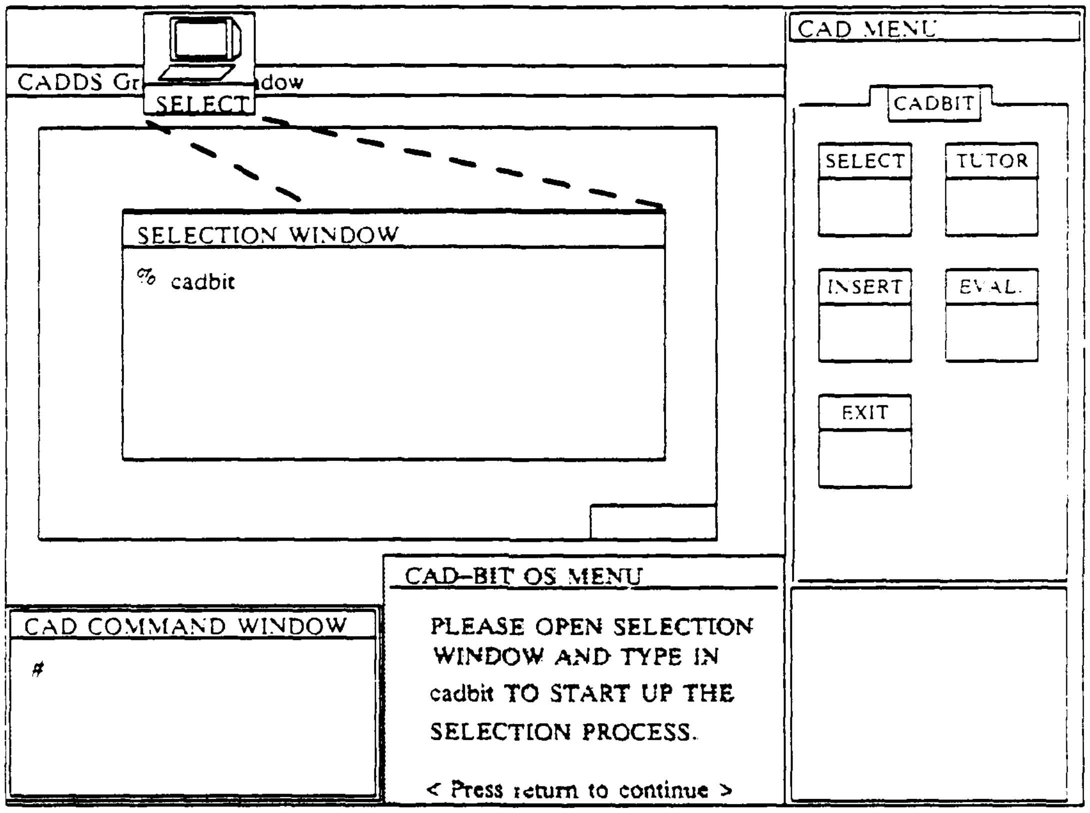


Figure 3-19
CAD-BIT OPERATING MODE: CHOOSING OPERATING SYSTEM ENVIRONMENT SELECTION PROCESS



PLEASE OPEN SELECTION WINDOW AND TYPE IN cadbit TO START UP THE SELECTION PROCESS.

< Press return to continue >

Figure 3–20
OPEN SELECTION WINDOW IN OPERATING SYSTEM ENVIRONMENT

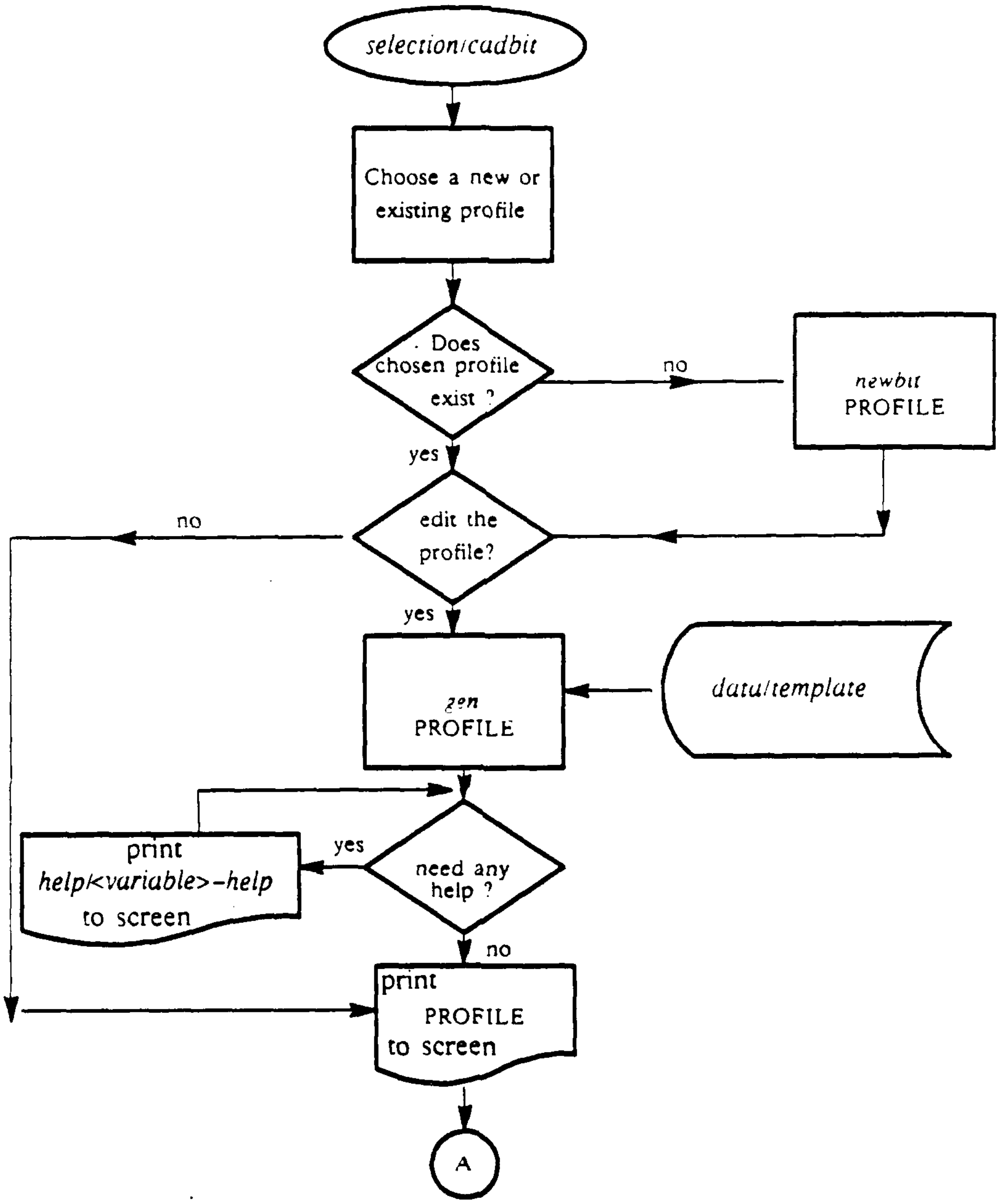


Figure 3-21
PROFILE GENERATION FLOW DIAGRAM

tory. If it does not exist, the C program *selection/newbit.c* is called to create the necessary directories, prompt the user for a brief description of the profile (user name, nomenclature, etc.) and copy a default profile into the new profile. Once the profile has been found or created, its full name is entered into the file *.cadbit* along with the command that sets up environment variables. Environment variables are those variables which, once initialized, are active for an entire session. From this point on the environment variable, PROFILE, will represent the current profile name. This is done so that after each login the name of the last profile worked on will be present in the UNIX environment without the user having to identify it when using the most recent selection. The user is then requested whether or not the profile is to be modified. If not, the profile is printed to the screen and stays as is. If the file is to be overwritten, the C program *selection/gen.c* is called and the data file *data/template* is read in. This file contains all the attributes and penalty parameters used for suitability, selection, and calculations. As each line of the file is read, the question and the default answer are printed to the screen. The user types in an answer for each question printed. See questions (SQS) in the Template file, Figure 3-48. An answer can consist of the following :

a. Question Mark (?) which causes the on-line documentation (help facilities) to be printed to the screen for a particular PCB attribute or penalty parameter. Each help file has the naming convention, *help/<variable>-help*. See section 3.9.1.2 for a listing of all the help file contents.

b. Exclamation Point (!) which causes the procedure to accept all of the following answers as defaults.

c. Carriage Return with no other input which causes the default answer for the particular attribute or penalty parameter to be accepted.

d. A character or string is taken as an answer but first it is checked for correctness by the C program, *selection/syntax.c*. The user's answer is passed along with the allowable answers to the syntax.c program and compared. If the user's answer matches one of the allowable answers, it is accepted and entered along with its attribute/parameter variable into the profile. If no match is found, the user is informed of the error and the corresponding help file is printed to the screen. The user is then prompted for a new answer. When the profile is complete, it is printed to the screen for verification. If something needs to be changed, the user re-runs the procedure.

(3) Outputs

The output of this procedure is a completed User Design Profile under the name *profile/<profilename>/<profilename>.cut*. In this file are contained the Suitability Attribute List (SAL), the Suitability Answer Set (SAS), the Penalty Parameter List (PPL), the Penalty Weighting Factors (PWF), and the Technique Override List (TOL). For a definition of these parameters, see section 7.1.

3.4.5.2 Suitability Determination

In this section a list of suitable BIT techniques is generated for a particular PCB design.

(1) Inputs

In generating a list of suitable BIT Techniques, three or more files are input into this procedure. The first, *data/technique-list*, is a list of all the BIT techniques available . The second file is the profile which contains the SAL and SAS for the particular PCB design. The third (or more) file(s), *<t##>/teq-file*, are the BIT technique attribute files themselves. Contained in each is the SAL and SAS which define suitability requirements for the BIT technique(s).

(2) Processing

The C program *selection/suitable.c*, Figure 3-22, obtains the environment variable, PROFILE, which contains the name of the most recently used profile. To determine the suitability of a technique the SAS from the profile is compared with the SAS of each BIT technique file, *<t##>/teq-file*. The data file, *data/technique-list*, is used to obtain the existing technique numbers for the techniques being compared. If all the applicable attributes in a technique file match those of the profile, then a technique is considered suitable and is entered into the file *data/<profilename>/suitable* otherwise it is entered into the file *data/<profilename>/rejects* along with the attribute variable that it failed on. All attributes may not be applicable for each technique. This will be indicated in the *<t##>/teq-file(s)* using the attribute value "@". Once the suitable techniques have been established, they are printed to the screen and the user is prompted as to whether or not the penalty calculations are to be generated. If not, the user has the option to view the rejects contained in the data file, *data/<profilename>/rejects*. The user is then prompted as to whether or not there are any forced exclusions and/or inclusions. If so, the C program, *selection/override.c*, is called which prompts the user for the technique numbers of the techniques to be excluded and/or included. Once complete, the new list of suitable techniques is generated *(data/<profilename>/suitable)* and the user is again prompted for penalty calculations.

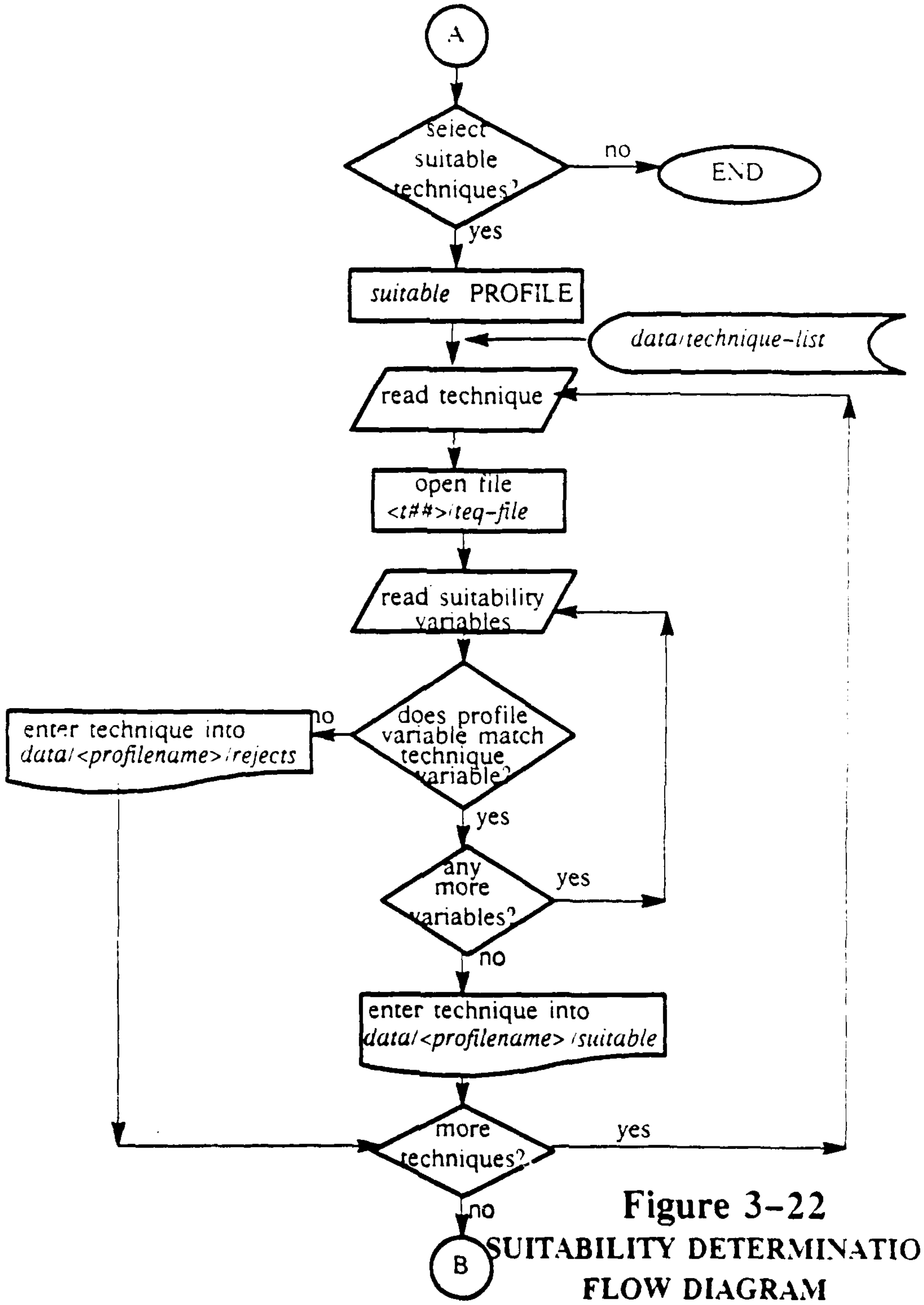

A
select suitable techniques?
no
END
yes
suitable PROFILE
data/technique-list
read technique
open file <t##>/teq-file
read suitability variables
does profile variable match technique variable?
no
enter technique into data/<profilename>/rejects
yes
any more variables?
yes
no
enter technique into data/<profilename>/suitable
more techniques?
yes
no
B

Figure 3-22
SUITABILITY DETERMINATION FLOW DIAGRAM

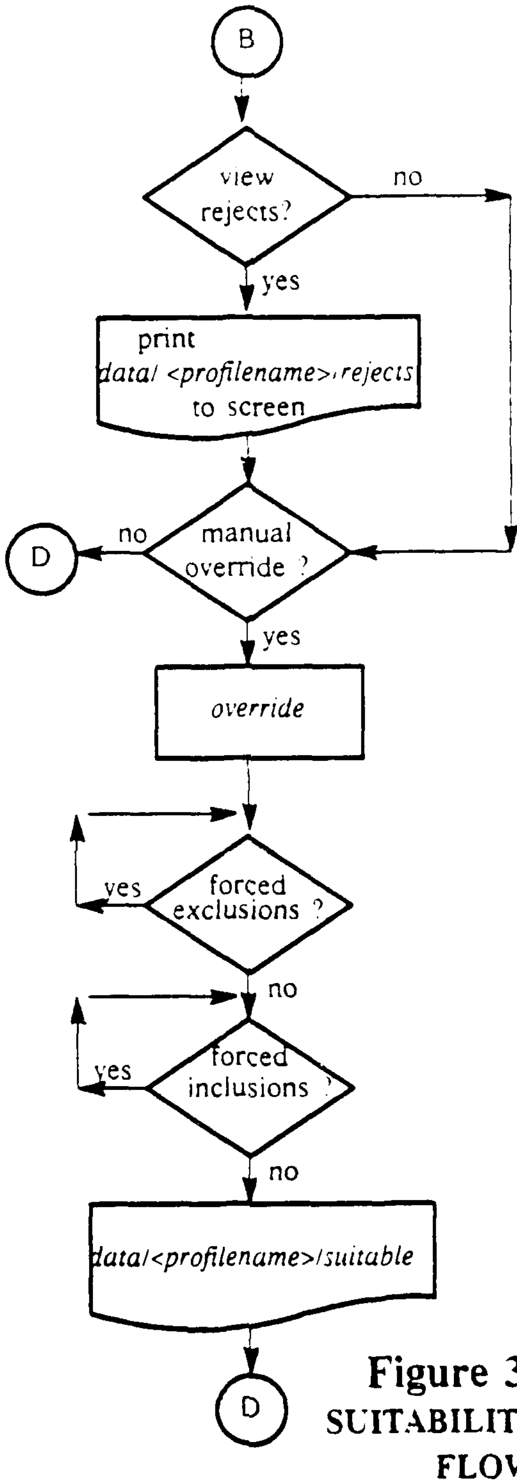


Figure 3-22 (continued)
SUITABILITY DETERMINATION FLOW DIAGRAM

(3) Outputs

Two files are generated from this procedure. The first, *data/<profilename>/suitable*, contains a list of suitable BIT techniques for a particular User Design Profile, Figure 3-23. The second, *data/<profilename>/rejects*, contains a list of all the BIT techniques rejected along with the attribute(s) the comparison failed on, Figure 3-24. When displayed, these files can be viewed in the Selection Window.

3.4.5.3 Technique Ranking (Penalty Calculations)

In this section, the suitable techniques are ranked in decreasing order of preference based upon penalty formulas associated with each BIT technique. These formulas can be found in Volume II, Data Base of BIT Functions. See Appendix-B for a listing of the penalty algorithms.

(1) Inputs

The file *data/<profilename>/suitable* that was generated in section 3.4.5.2 is input into this procedure along with the profile which contains the weighting factors that are to be applied. For each suitable technique, the user is prompted for information pertaining to the technique.

(2) Processing

The C program *selection/penalty.c*, Figure 3-25, reads in the profile along with the corresponding list of suitable techniques, *data/<profilename>/suitable*. The technique numbers and names are read in from the list. The technique numbers are tested in a C switch statement. For each case, a subroutine with the name *<t##>_form* is called and the penalty parameters which are represented as C structure pointer variables, as well as a character string containing the profile name, are passed to it. The file name structure of these C programs is *<t##>/formulas.c*. Each of these routines is generated through the *fix-penalty* utility. Through these C programs the user will be prompted for User Requested Data (URD - input pins, output pins, etc.) which can be found in the Volume II, Data Base of BIT Functions. The user's answers are placed in a file, *profile/<profilename>/<t##>-urd*, for future use. If this file exists upon execution of this procedure, the user is prompted as to whether or not the original answers are to be accepted or disregarded. The answers are then applied to Component Determination Equations (CDEs). These equations generate the quantity of circuits necessary for the particular PCB design. The results of these calculations and the names of the corresponding circuits

T01	ON-BOARD ROM
T02	PSEUDO RANDOM PATTERN GENERATOR/ MULTIPLE INPUT SHIFT REGISTER (PRPG/MISR)
T03	MICROPROCESSOR BIT
T08	BUILT-IN LOGIC BLOCK OBSERVER (BILBO)

SAMPLE FILE CONTAINING SUITABLE TECHNIQUES.

Figure 3-23
TECHNIQUE SUITABILITY

T03	MICROPROCESSOR BIT REJECTED DUE TO ATTRIBUTE USAGE. NO MATCH BETWEEN USAGE IN TECHNIQUE (SAA) AND USAGE IN PROFILE.
T04	ON-BOARD INTEGRATION OF VLSI CHIP BIT (OBIVCB) REJECTED DUE TO ATTRIBUTE MICROPROCESSOR. NO MATCH BETWEEN MICROPROCESSOR IN TECHNIQUE (SAA) AND MICROPROCESSOR IN PROFILE.

SAMPLE FILE CONTAINING BRIEF EXPLANATION AS TO WHY A TECHNIQUE WAS REJECTED DURING THE SUITABILITY TEST PHASE.

Figure 3-24
TECHNIQUE REJECTION

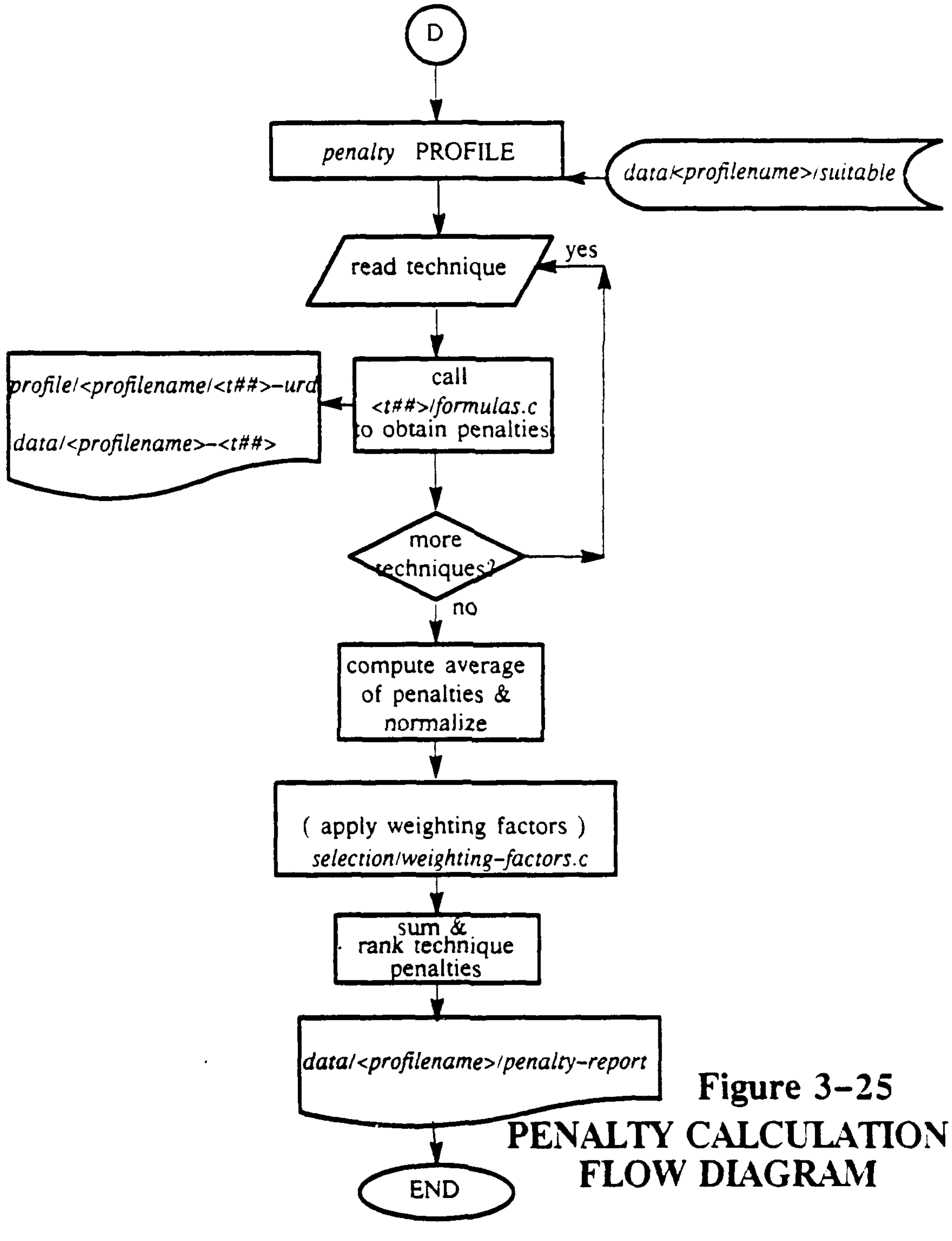


Figure 3-25
PENALTY CALCULATION FLOW DIAGRAM

are placed in a file, *data/<profilename>-<t##>*, Figure 3-26. This data is later used in the BIT Insertion section to determine how many of each circuit type is to be placed in the PCB. The quantity calculated is then used in the Technique Penalty Equations (TPEs). These penalties are stored dynamically in the structure pointer variables that were passed to the subroutine.

After control is passed back to *selection/penalty.c*, an internal subroutine (*normalize*) is called to obtain the average penalty for each Technique Penalty Parameter (TPP). These averages are then normalized through another formula. Algorithms 4 and 5 of Appendix-B are implemented to achieve this.

After normalization, the subroutine *selection/weighting-factors.c* is called to apply the profile weighting factors to the normalized array. Refer to algorithm 6 of Appendix-B. If the penalties in the profile (PPL) don't match the penalties in the array, a warning which contains the name of the missing penalty parameter is printed. The resulting penalties are placed in the penalty report. Each technique is then ranked by adding the penalties for each technique together and comparing the sums. These Figures of Merit (FOM) are calculated through the internal subroutine, *rank_penalty* (algorithm 7 of Appendix-B). The techniques and their penalties are then sorted in decreasing order of preference with the best technique having the lowest penalty ranking. This information is then placed in the penalty report, *data/<profilename>/penalty-report.*

(3) Outputs

The file *data/<profilename>/penalty-report* is output to the Selection Window. This file is a ranked order list of BIT Techniques with associated penalties. They are ranked in decreasing order of preference, Figure 3-27. There are individual lists that are output, *data/<profilename>-<t##>*, Figure 3-26, which contain two columns of information : the circuit name and the quantity of circuit elements which are to be placed in the PCB. User Requested Data files are also output. They contain the user's responses to the User Requested Data for each suitable technique.

3.4.6 BIT TECHNIQUE INSERTION

The BIT Insertion process is Function 5 of the main operating system (UNIX) menu. The result of this menu choice is an execute file to be used in the CAD environment to insert specific entities into the PCB. Prior to this, however, one is advised to enter the BIT Technique Insertion Diagram (BTID) for the technique being implemented into the PCB.

data/<profilename>–<t##>

CIRCUIT	QUANTITY
ROM	4
MUX	1
●	●
●	●
●	●
COMPARATOR	1

Figure 3–26
CIRCUIT QUANTITY FILE

data/<profilename>/penalty-report

```
                         CAD--BIT
                   TECHNIQUE SELECTION

                                                              TEST
    TECHNIQUE              AREA     POWER     WEIGHT          TIME    RANKING
ID  DESCRIPTION            mm2      mw        grams           usec    %
--  -------------          ------   --------  --------        -----   --------

t01 On-Board ROM           4.35     5010      46               195    44

t02 PRPG/MISR              4.82     5825      51              1820    68
                                                     • • •
t08 BILBO                  7.74     4612      32              1650    71

t03 Microprocessor BIT     2.91     3720      35              2113    84
--  -------------          ------   --------  --------        -----   --------


Suitable BIT techniques are ranked in decreasing order of preference.
```

Figure 3-27
PENALTY REPORT

section 3.9.2.7. The BTID guides the insertion process and provides a check-off mechanism to keep track of inserted BIT circuits (Insert Circuit, BIT Flag Insertion and BIT Group Insertion).

3.4.6.1 Insert Circuit ("Do" Command Generation & Implementation)

The "DO" commands, Figure 3-28, are generated execute files which contain the host CAD environment commands for the insertion of circuits into the PCB. The commands include the number and name(s) of the circuit(s) to be inserted.

(1) Inputs

Five files are input into this procedure. The first, *data/technique-list*, contains the list of BIT techniques and their respective technique numbers. The second, *<t##>/circuitmenu*, contains a list of all circuits and the corresponding default parts used by the particular technique. The third, *data/default-parts*, contains all the library names and default part numbers for circuits to be inserted into the PCBs. The fourth, *data/host-commands*, contains a list of graphics "insert component" commands for different CAD systems. The fifth, *data/<profilename>-<t##>*, contains a list of circuits and the quantities of each that are to be inserted into the PCB design for a particular BIT Technique. This file is generated in the Penalty calculation section of the Selection procedure.

(2) Processing

The C-Shell script *os-menu/insert-bit*, Figure 3-29, prints the file, *data/technique-list*, to the screen and prompts the user for the technique number of the BIT technique that is to be implemented. The file, *<t##>/circuitmenu*, is then printed to the screen which contains the list of circuits available for the selected technique. The technique number and circuit name chosen are passed to the C program *bit-ins/search.c* which creates an execute file, *<t##>/insert/<circuit>*, to insert the calculated number of circuits into the PCB design, Figure 3-28. The C program first obtains the profile name through the environment variable, PROFILE. The file *data/<profilename>-<t##>* is accessed and the circuit name that was selected is compared with the circuit names in the file. When a match is found, the quantity associated with the circuit is read in. The file *<t##>/circuitmenu* is then open. The circuit names are compared and, when a match is found, the default part number is read in. The circuit name and default part number are compared to the information in the data file, *data/default-parts*, to find the CAD library name of the circuit. The appropriate insertion command for a CAD system is located on the first line of *data/host-commands*. The quantity, library name, and CAD command are then entered into the execute file as an insertion command. This insertion command is executed via the CAD graphics menu and results in the BIT circuits being inserted in the PCB. Figures 3-30 through 3-36 illustrate the BIT insertion procedure.

(3) Outputs

The number and name of the circuit(s) to be inserted is printed to the MCBW. The graphics command file *<t##>/insert/<circuit>* is also created containing a line such as : Insert circuit(s) "circuit name" location "p1,p2...", which is used at CAD graphics level to insert the actual circuits into the PCB.

t01/insert/rom

SCREEN OUTPUT

```
Using 4 1024X8 ROMS P/N TBP38516
PLEASE DIGITIZE 4 LOCATIONS
```

RESULT

GENERATES COMMAND IN HOST CAD SYNTAX TO INPUT 4 1024X8 ROMS OF DEFAULT PART NUMBER TBP38516.
IN COMPUTERVISION CADDS 4X SYNTAX:

```
INSERT NFIGURE CADBIT.TBP38516: MODEL LOC dddd
```

Figure 3-28
"DO" COMMANDS

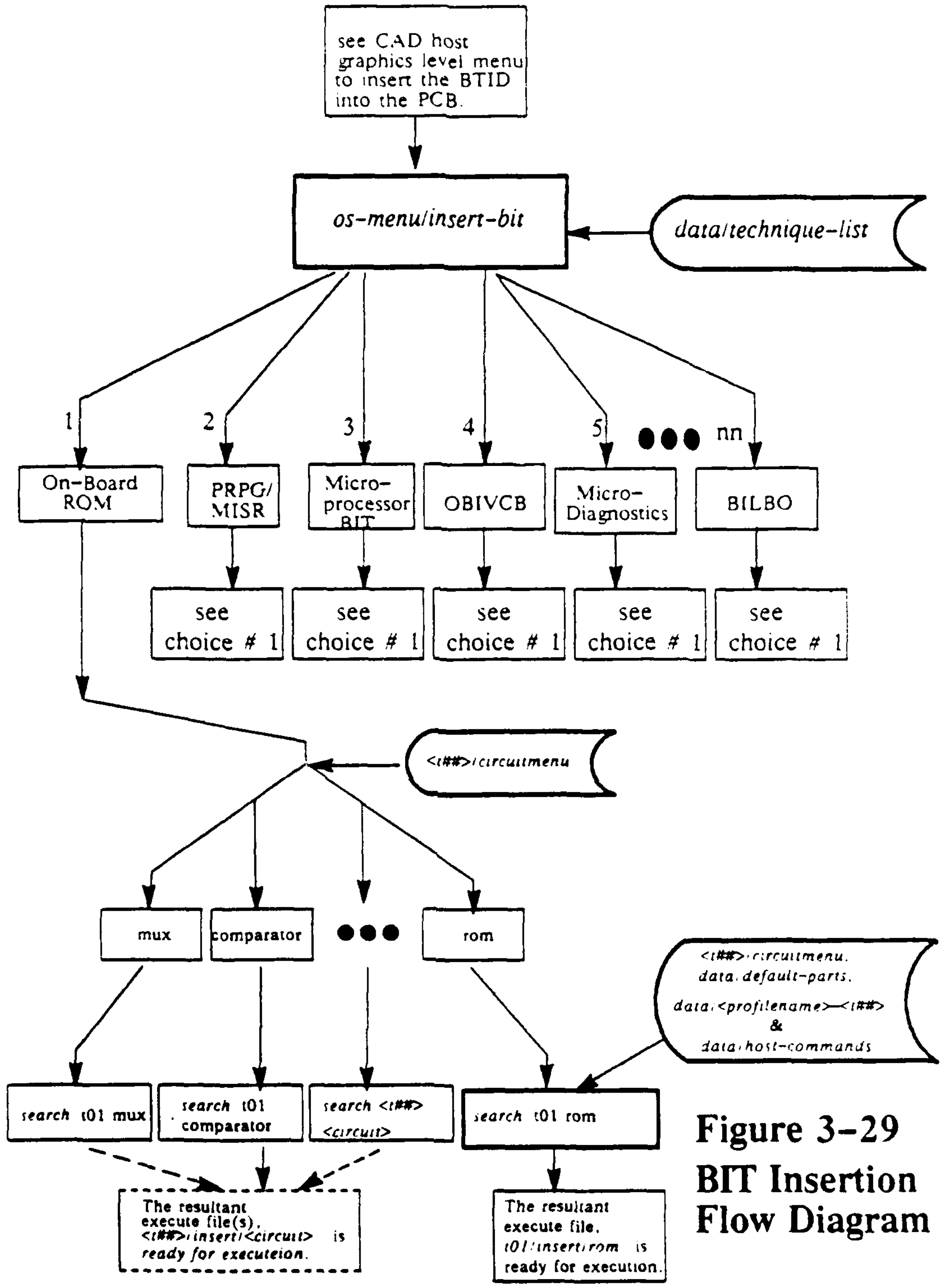
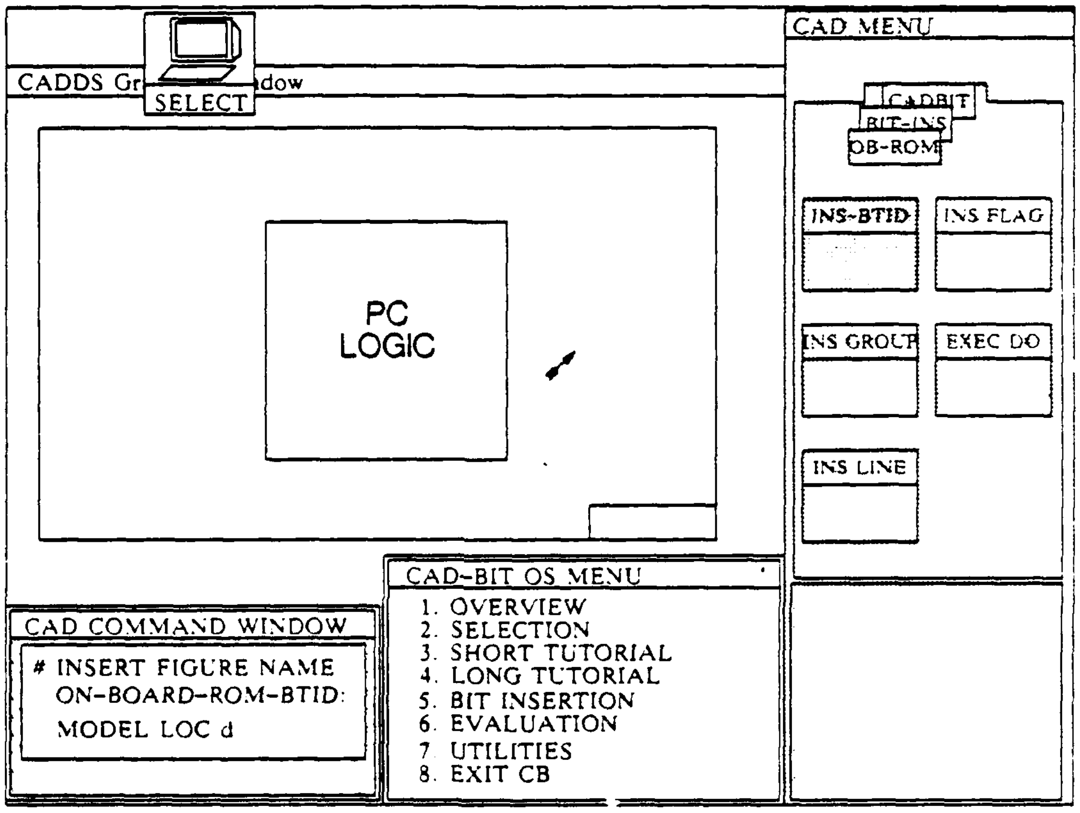

see CAD host graphics level menu to insert the BTID into the PCB.
os-menu/insert-bit
data/technique-list
1
2
3
4
5
nn
On-Board ROM
PRPG/ MISR
Micro-processor BIT
OBIVCB
Micro-Diagnostics
BILBO
see choice # 1
see choice # 1
see choice # 1
see choice # 1
see choice # 1
<t##>/circuitmenu
mux
comparator
rom
<t##>/circuitmenu, data/default-parts, data/<profilename>—<t##> & data/host-commands
search t01 mux
search t01 comparator
search <t##> <circuit>
search t01 rom
The resultant execute file(s), <t##>/insert/<circuit> is ready for executeion.
The resultant execute file, t01/insert/rom is ready for execution.

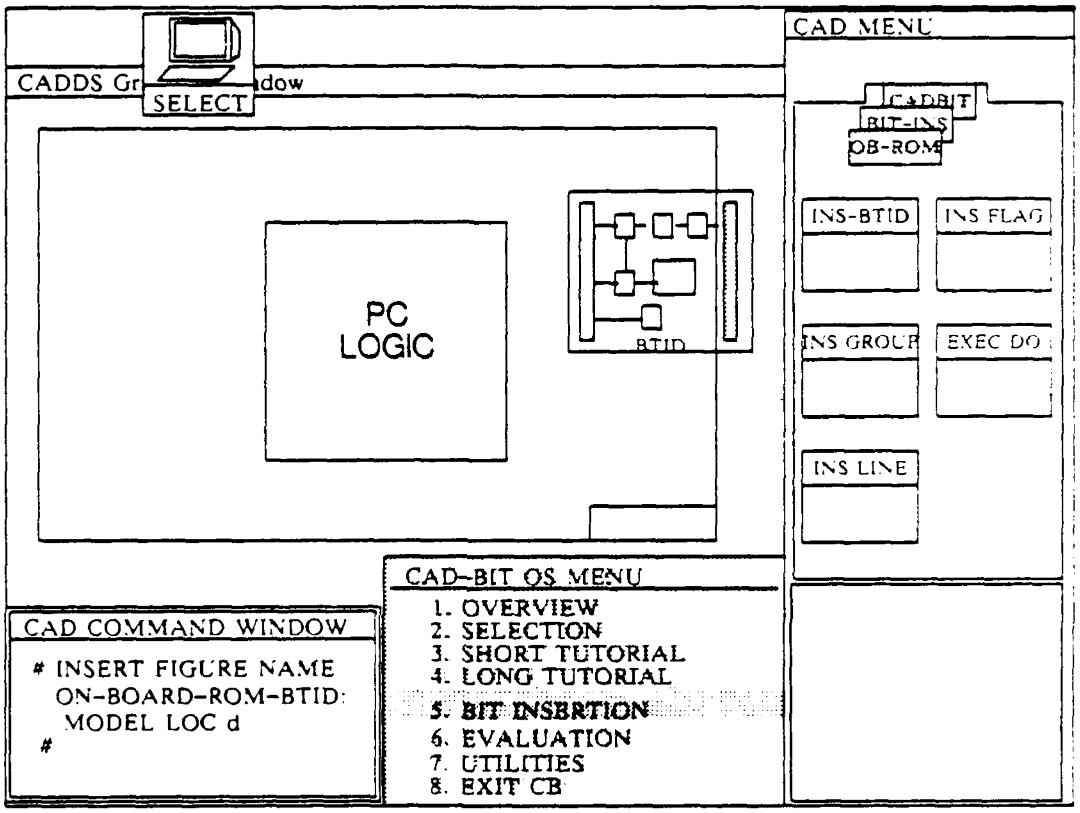
Figure 3-29 BIT Insertion Flow Diagram



1. OVERVIEW
2. SELECTION
3. SHORT TUTORIAL
4. LONG TUTORIAL
5. BIT INSERTION
6. EVALUATION
7. UTILITIES
8. EXIT CB

```
# INSERT FIGURE NAME
  ON-BOARD-ROM-BTID:
  MODEL LOC d
```

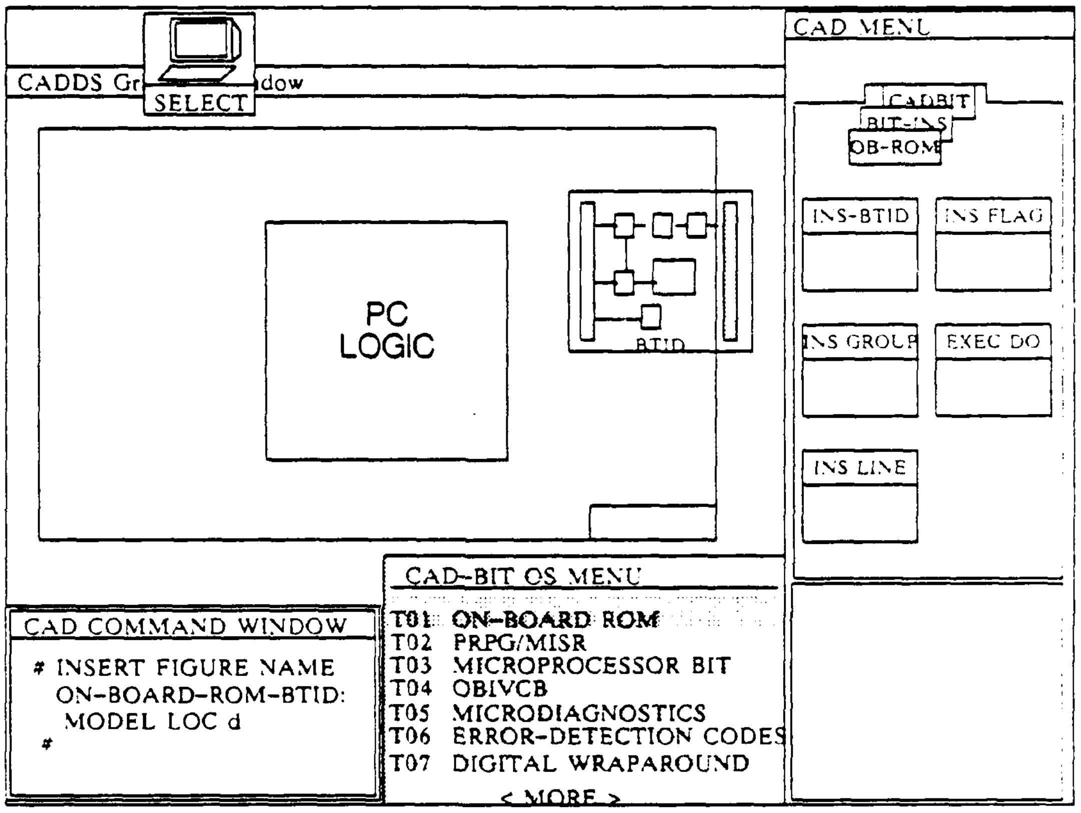
Figure 3-30
BTID INSERTION SELECTION IN CAD ENVIRONMENT



```
# INSERT FIGURE NAME
  ON-BOARD-ROM-BTID:
  MODEL LOC d
#
```

Figure 3–31

BTID INSERTED IN PCB DRAWING SPACE IN CAD ENVIRONMENT: OS MENU BIT INSERTION SELECTED

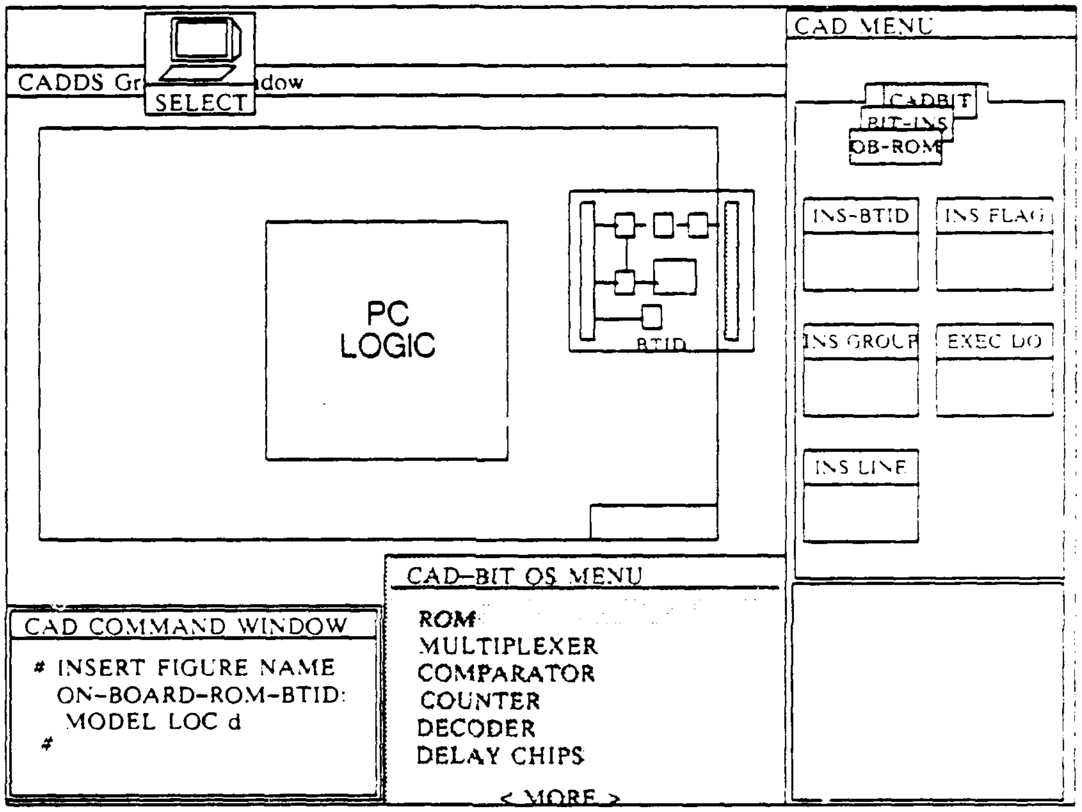


```
# INSERT FIGURE NAME
  ON-BOARD-ROM-BTID:
   MODEL LOC d
 #
```

```
T01  ON-BOARD ROM
T02  PRPG/MISR
T03  MICROPROCESSOR BIT
T04  OBIVCB
T05  MICRODIAGNOSTICS
T06  ERROR-DETECTION CODES
T07  DIGITAL WRAPAROUND
         < MORE >
```

Figure 3-32

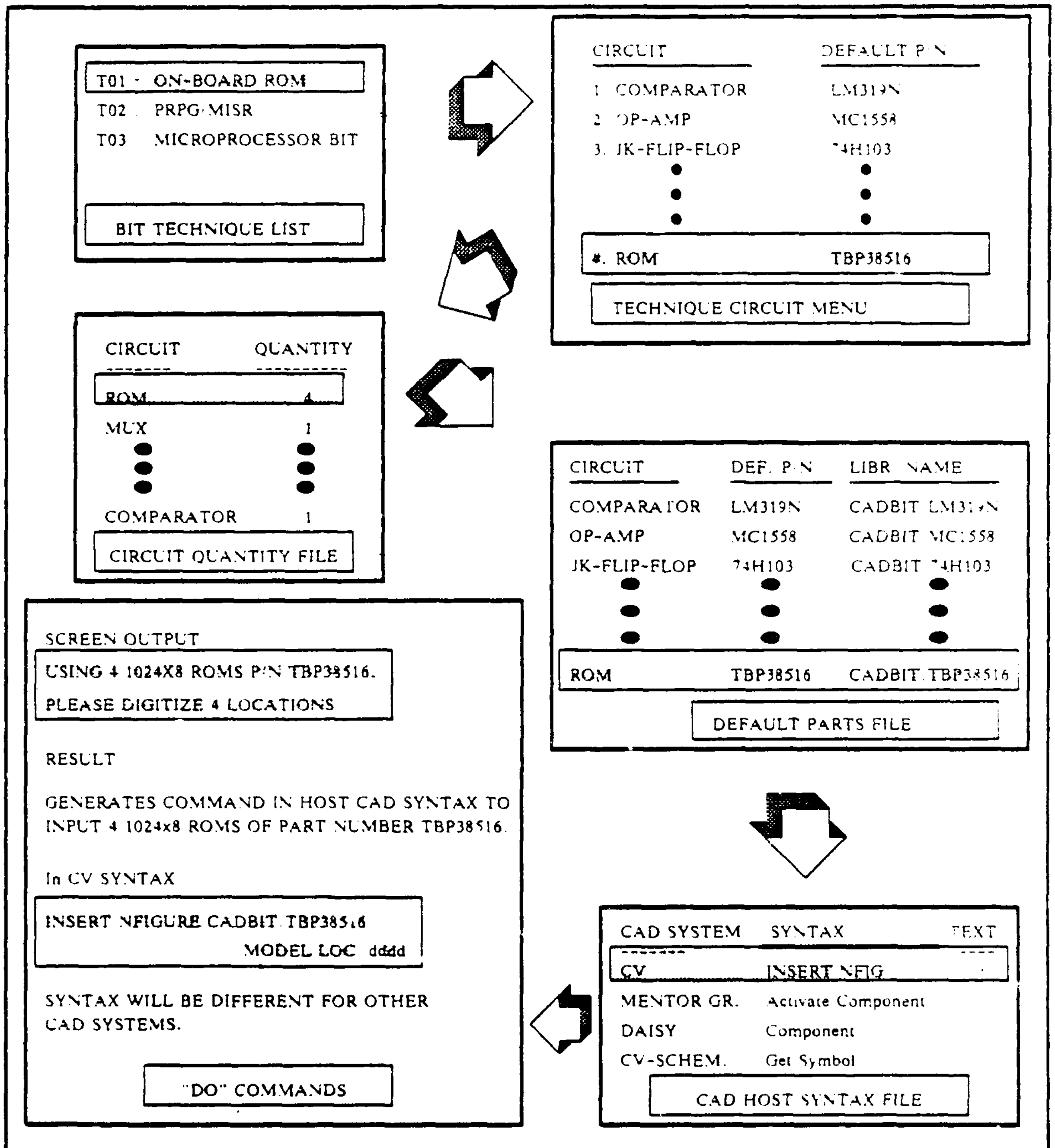
OS MENU BIT INSERTION: TECHNIQUE SELECTED



```
# INSERT FIGURE NAME
  ON-BOARD-ROM-BTID:
   MODEL LOC d
 #
```

Figure 3–33

OS MENU BIT INSERTION: CIRCUIT SELECTION



T01 -	ON-BOARD ROM
T02	PRPG/MISR
T03	MICROPROCESSOR BIT

BIT TECHNIQUE LIST

CIRCUIT	DEFAULT P/N
1. COMPARATOR	LM319N
2. OP-AMP	MC1558
3. JK-FLIP-FLOP	74H103
●	●
●	●
●	●
#. ROM	TBP38516

TECHNIQUE CIRCUIT MENU

CIRCUIT	QUANTITY
ROM	4
MUX	1
●	●
●	●
●	●
COMPARATOR	1

CIRCUIT QUANTITY FILE

CIRCUIT	DEF. P/N	LIBR NAME
COMPARATOR	LM319N	CADBIT LM319N
OP-AMP	MC1558	CADBIT MC1558
JK-FLIP-FLOP	74H103	CADBIT 74H103
●	●	●
●	●	●
●	●	●
ROM	TBP38516	CADBIT TBP38516

DEFAULT PARTS FILE

CAD SYSTEM	SYNTAX	TEXT
CV	INSERT NFIG	
MENTOR GR.	Activate Component	
DAISY	Component	
CV-SCHEM.	Get Symbol	

CAD HOST SYNTAX FILE

SCREEN OUTPUT

```
USING 4 1024X8 ROMS P/N TBP38516.
PLEASE DIGITIZE 4 LOCATIONS
```

RESULT

GENERATES COMMAND IN HOST CAD SYNTAX TO INPUT 4 1024x8 ROMS OF PART NUMBER TBP38516.

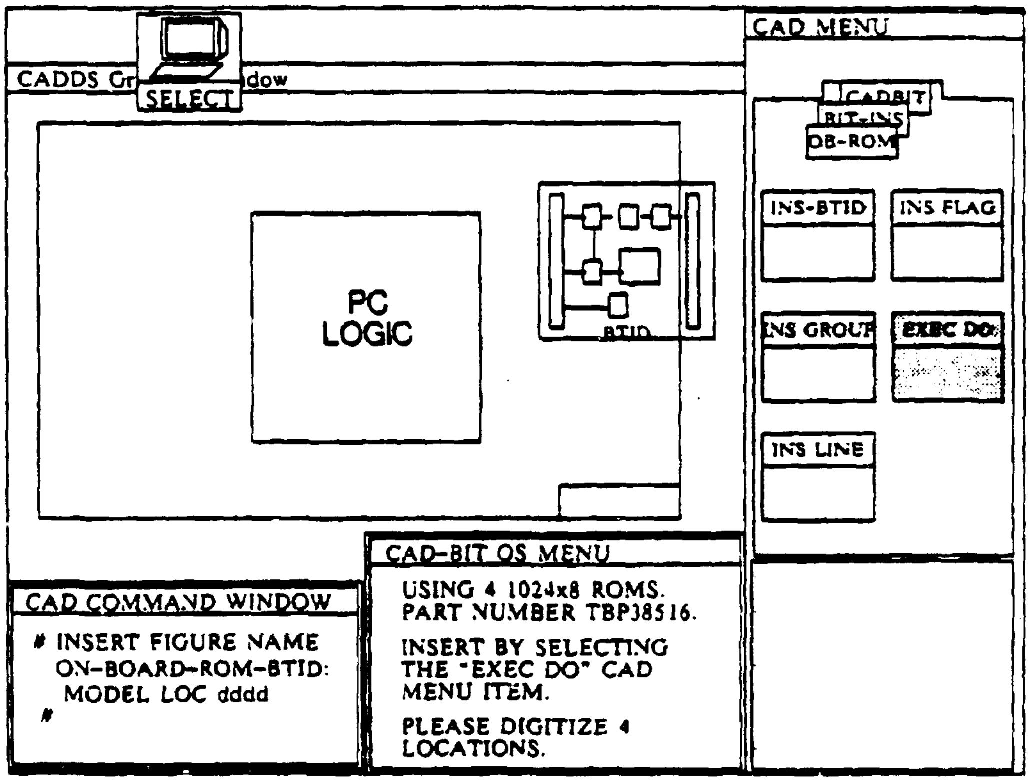
In CV SYNTAX

```
INSERT NFIGURE CADBIT TBP38516
                    MODEL LOC  dddd
```

SYNTAX WILL BE DIFFERENT FOR OTHER CAD SYSTEMS.

"DO" COMMANDS

Figure 3-34
"DO" COMMAND GENERATION



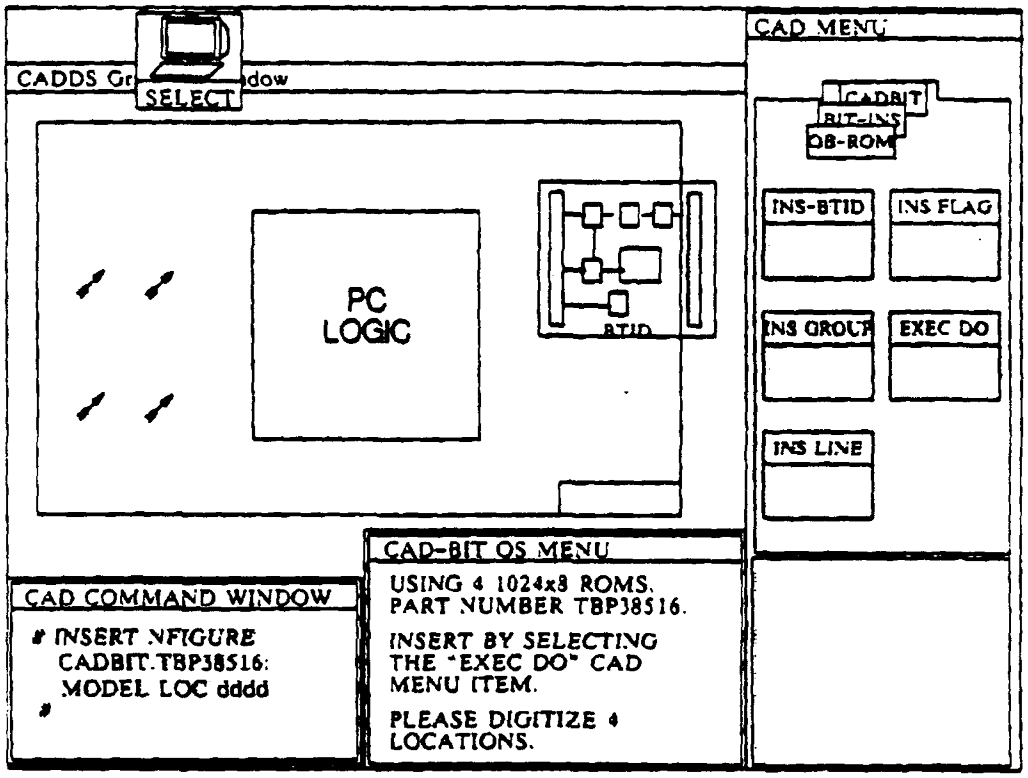
```
# INSERT FIGURE NAME
  ON-BOARD-ROM-BTID:
   MODEL LOC dddd
 #
```

USING 4 1024x8 ROMS.
PART NUMBER TBP38516.

INSERT BY SELECTING THE "EXEC DO" CAD MENU ITEM.

PLEASE DIGITIZE 4 LOCATIONS.

Figure 3–35
CAD ENVIRONMENT COMMAND EXECUTION



# INSERT NFIGURE
CADBIT.TBP38516:
MODEL LOC dddd
#

USING 4 1024x8 ROMS.
PART NUMBER TBP38516.
INSERT BY SELECTING THE "EXEC DO" CAD MENU ITEM.
PLEASE DIGITIZE 4 LOCATIONS.

Figure 3–36
CAD MENU "DO" COMMAND EXECUTION

3.4.6.2 BIT Flag Insertion (CAD mode)

The BIT flag is a property/attribute with an associated value to be assigned to each BIT Circuit Component (i.e. each circuit component which is BIT related). This information will be extracted by the host system data extraction procedure for the Evaluation routine.

(1) Inputs

The designer enters the property/attribute "BITFLAG" and a value (real decimal value in the range of 0.0 to 1.0) to indicate that the component is part of the BIT circuitry. The decimal value indicates the fraction of the component that is allocated to BIT.

(2) Processing

A property containing the attribute and the real decimal value (0.0 to 1.0) is entered onto each BIT Circuit Component in the PCB through the use of a host graphics level command. This command is contained in the graphics menu or it can be typed in. Figure 3-37 illustrates the insertion of the BIT flag attribute.

(3) Outputs

The data base will be updated with the BITFLAG attribute and the associated value.

3.4.6.3 BIT Group Insertion (CAD mode)

This will mark the component as BIT circuitry and place a BIT group number on the components. This information will be extracted by the host system data extraction procedure for the Evaluation routine.

(1) Inputs

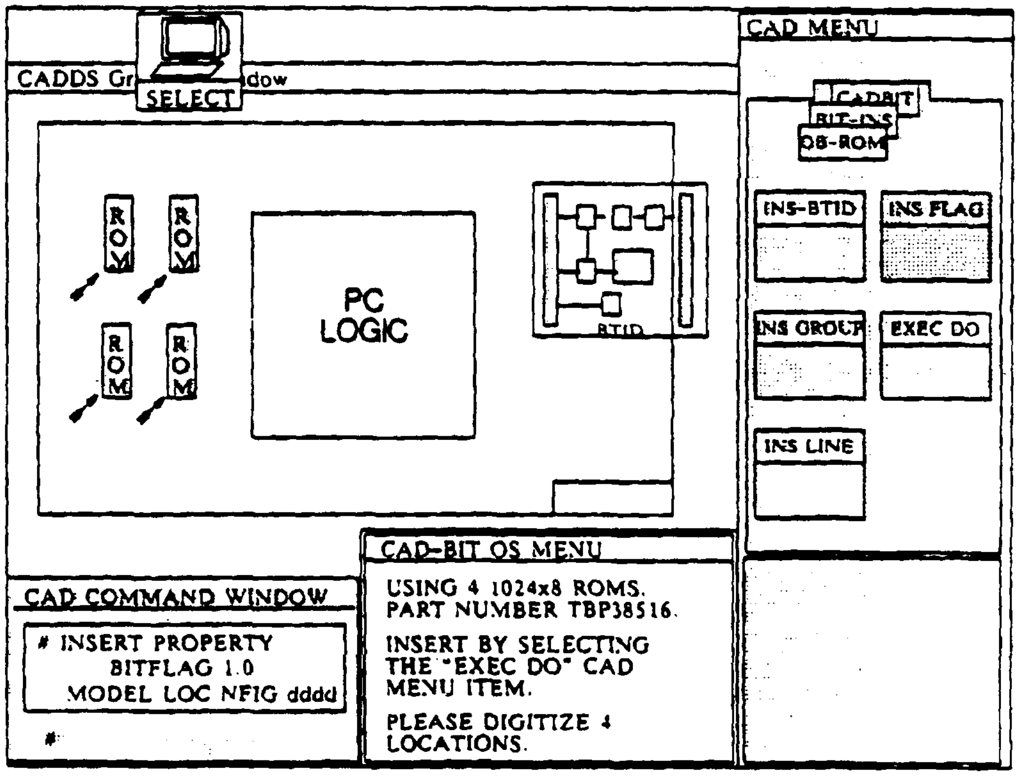
The designe. enters the property/attribute "BITGROUP" and a value (integer from 1 to nn) to indicate the BIT group that the component belongs to.

(2) Processing

A property containing the attribute and integer value representing the BIT group number is entered onto each BIT Circuit Component in the PCB through the use of a host graphics level command. This command is contained in the graphics menu or it can be typed in. Figure 3-38 illustrates the insertion of the BIT group attribute.

(3) Outputs

The data base will be updated with the BITGROUP attribute and the associated value.



# INSERT PROPERTY
BITFLAG 1.0
MODEL LOC NFIG dddd
#

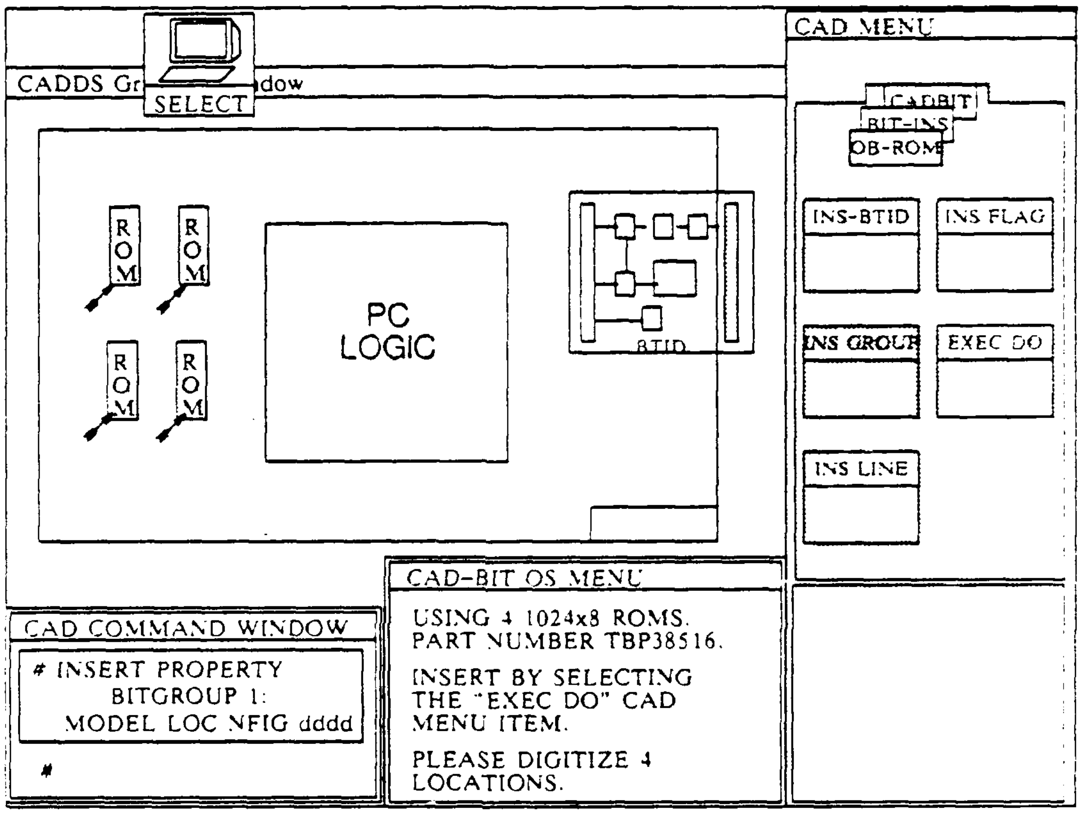
USING 4 1024x8 ROMS.
PART NUMBER TBP38516.

INSERT BY SELECTING THE "EXEC DO" CAD MENU ITEM.

PLEASE DIGITIZE 4 LOCATIONS.

Figure 3–37

**ROMS INSERTED.
SELECT CAD MENU
INSERT BIT FLAG ATTRIBUTE ICON**



```
# INSERT PROPERTY
      BITGROUP 1:
   MODEL LOC NFIG dddd

#
```

USING 4 1024x8 ROMS.
PART NUMBER TBP38516.

INSERT BY SELECTING
THE "EXEC DO" CAD
MENU ITEM.

PLEASE DIGITIZE 4
LOCATIONS.

Figure 3-38
SELECT CAD MENU
INSERT BIT GROUP ATTRIBUTE ICON.

3.4.7 BIT TECHNIQUE EVALUATION

3.4.7.1 BIT Penalty Data Extraction From CAD Schematic

It is assumed that a schematic design data extraction language is available on the host CAD system.

(1) Inputs

Inputs will have been provided during the BIT Technique Insertion phase including the BIT FLAG and BIT GROUP attributes and values.

(2) Processing

A host schematic graphics data base extract program is used to extract the part information from the PCB schematic. These programs are specific to each host system (Computervision, Mentor, Daisy, etc). It is the responsibility of the user or CAD support group to write these programs.

(3) Outputs

A data file containing all the part information extracted from the PCB is generated under the name *data/cadbit-eval-data*. It is required that the output format be as described in Figure 3-39.

3.4.7.2 BIT Ranking

The CAD-BIT Ranking procedure is Function 6 of the main operating system (UNIX) menu. This procedure is used in conjunction with section 3.4.7.1 to generate a ranking of the BIT circuitry actually placed into the PCB.

(1) Inputs

Three files are input into this procedure: the Parts Penalty Data file, *data/ppd*, the profile, and the file which results from the extract procedure on the CAD schematic, Figure 3-39.

(2) Processing

The C program, *eval/cadbit-eval.c*, reads in the above three files, *data/ppd*, the profile, and *data/cadbit-eval-data*. For each circuit in the file *data/cadbit-eval-data*, the BIT factors found in *data/cadbit-eval-data* are applied to each penalty parameter in the file *data/ppd* to arrive at an actual set of penalties for the circuit. Each of these penalty parameters is compared against those in the current profile to find any inconsistencies in the data.

data/cadbit-eval-data

Element	BIT factor	Partno	BIT group
U1	0.50	LM319N	1
U2	0.50	74H103	2
●	●	●	●
●	●	●	●
●	●	●	●
Un	0.50	MC1558	3
R11	0.30	N/A	4
R12	0.20	N/A	4
Rnn	0.10	N/A	5

This is a sample data file which results from the host graphics extract routine on the PCB during the Evaluation Procedure.

Figure 3-39

DATA EXTRACTION REPORT

If there is a discrepancy, a warning will be printed which specifies the missing penalty parameter. After all the BIT factors have been applied, each of these parameters is then summed based upon the BIT group that the circuits belong to. Their groupings can be found in *data/cadbit-eval-data*. The results are then placed in the file *data/<profilename>/btpr*, Figure 3-40. Figure 3-41 illustrates the generation of the BIT Technique Penalty Report. See Appendix-C for the Evaluation algorithms.

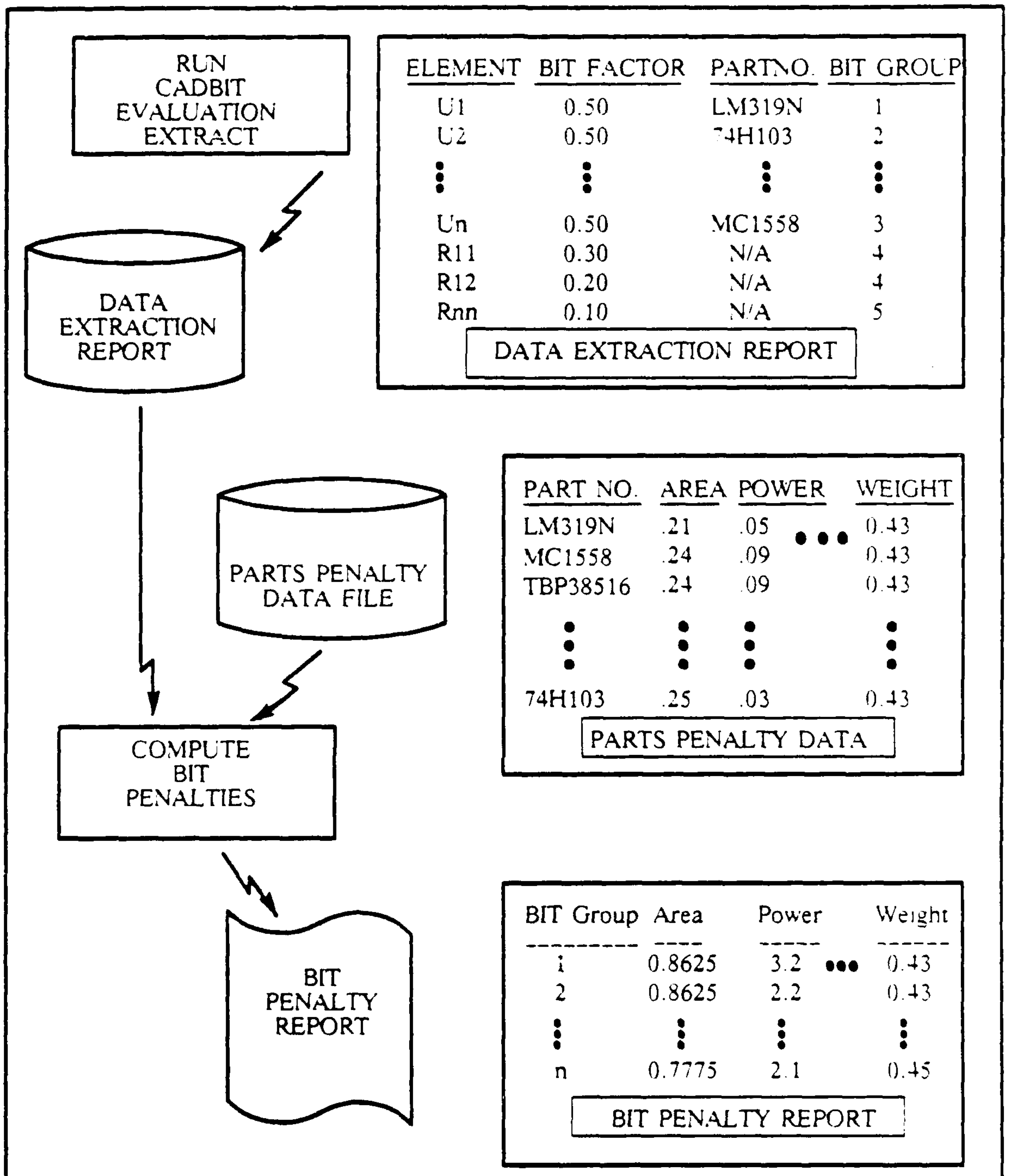
(3) Outputs

A BIT Technique Penalty Report, *data/<profilename>/btpr*, Figure 3-40, is generated as a result of the BIT Ranking procedure. This file, which contains the BIT group number along with the various penalties associated with it, is displayed in the MCBW.

```
                    Bit Penalty Report
                    ------------------

   BIT Group    Area          Power           Weight
   ---------    ----          -----           ------
       1        0.8625        3.2              0.43
                                      ●●●
       2        0.8625        2.2              0.43

       ●          ●            ●                ●
       ●          ●            ●                ●
       ●          ●            ●                ●

       n        0.7775        2.1              0.45
```

Figure 3-40

BIT TECHNIQUE PENALTY REPORT



ELEMENT	BIT FACTOR	PARTNO.	BIT GROUP
U1	0.50	LM319N	1
U2	0.50	74H103	2
⋮	⋮	⋮	⋮
Un	0.50	MC1558	3
R11	0.30	N/A	4
R12	0.20	N/A	4
Rnn	0.10	N/A	5

DATA EXTRACTION REPORT

PART NO.	AREA	POWER		WEIGHT
LM319N	.21	.05	•••	0.43
MC1558	.24	.09		0.43
TBP38516	.24	.09		0.43
⋮	⋮	⋮		⋮
74H103	.25	.03		0.43

PARTS PENALTY DATA

BIT Group	Area	Power		Weight
1	0.8625	3.2	•••	0.43
2	0.8625	2.2		0.43
⋮	⋮	⋮		⋮
n	0.7775	2.1		0.45

BIT PENALTY REPORT

Figure 3-41

BIT PENALTY EVALUATION

3.4.8 UTILITIES

The utilities are most often C-Shell scripts used to alter or create procedures when a new technique and/or attributes are added or deleted. The utilities are found in the main operating system (UNIX) Function 7. To access them, type in the appropriate command in the Selection Window. The three utilities are in the following subparagraphs (Add-Technique, Gen-Circuits and Fix-Penalty).

3.4.8.1 Add-Technique

The C-Shell script *utilities/add-technique* adds a new technique into the data file *data/technique-list*. Once the new technique is added, the C program *selection/create-teq.c* is run to generate a new BIT Technique file, *<t##>/teq-file*, which contains the technique identification, SAL and SAS for the new technique.

(1) Inputs

The user inputs the name of the new technique as well as the attributes associated with that technique.

(2) Processing

The utility *utilities/add-technique* checks for the next available technique number in *data/technique-list*, assigns that number to the technique, and adds the entry into *data/technique-list*. A new directory called *<t##>* is created under the home directory. The routine *selection/create-teq.c* is then run. This routine creates the data file, *<t##>/teq-file*, by reading the SAL, SAA, and SQS from *data/template*. Each question from the SQS is printed to the screen. For each question, the user types in an answer. In *selection/syntax.c* the answer is compared with the allowable answers (SAA) for that attribute in the SAL. If the user's answer matches one of the allowable answers, it is accepted and entered along with the attribute into *<t##>/teq-file*. If no match is found, the user is informed of the error and the corresponding help file is printed to the screen. The user is then prompted for a new answer. See Figure 3-48 for SAL, SAA, and SQS entries.

(3) Outputs

The data file *<t##>/teq-file* is generated, the technique subdirectory is created, and the technique list is updated as a result of this utility.

3.4.8.2 Gen-Circuit

The C-Shell script *utilities/gen-circuit* adds a new circuit and other pertinent information (library name and/or default part number) to the files *<t##>/circuitmenu* and *data/default-parts*.

(1) Inputs

The user inputs the technique number (*t##*) of the technique which the circuit is to be associated with as well as the Circuit name, the default part number, and CAD library name.

(2) Processing

The utility *utilities/gen-circuit* checks for the existence of the specified circuit in all the *<t##>/circuitmenu(s)* where it is to be placed. If the circuit already exists in any files, those files are not changed. The file(s), as well as *data/default-parts*, are then sorted in alphabetical order according to the Circuit names.

(3) Outputs

If the outputs do not already exist, *<t##>/circuitmenu* is created, and the *data/default-parts* file is updated.

3.4.8.3 Fix-Penalty

The C-Shell script *utilities/fix-penalty* generates a C program containing the penalty formulas for a particular technique with the naming convention *<t##>/formulas.c*. This utility also alters the C program *selection/penalty.c* to incorporate new technique(s). This utility generates C programs used in section 3.4.5.3.2. See Figures 3-42 and 3-43 for sample run of this utility.

(1) Inputs

The user inputs the technique number of the BIT technique being added and/or altered, if applicable, as well as any questions and formulas which will be used in the generation of the penalty report. This information is located in Volume II, Data Base of BIT Functions. The file *data/penalty-cases* is also input into this utility. This file contains a C switch statement block which, depending upon the techniques considered suitable, will branch to the specific C subroutines *<t##>/formulas.c* to perform the Technique Penalty Equations (TPEs).

(2) Processing

At the user's request, the following C programs can be altered : *<t##>/formulus.c*, *selection/penalty.c*, and *selection/weighting-factors.c*. The user inserts into this utility the technique number, which is used to create or modify the subroutine *<t##>/formulas.c*, and the questions and formulas to be used in the routine. The technique number is then used to

Sample Penalty Formula Generation WorkSheet For On-Board ROM Technique

PART 1A INTRODUCTION *****

1. Enter technique number (if applicable) :________________t01__________

If altering a TPE file :

2. How many User Requested Data questions (URDs) will be asked ?_____5_____

(see Data Base of BIT Functions for questions pertaining to particular technique (URDs))

3. How many Component Determination Equations (CDEs) will be used ?___8_____

(see Data Base of BIT Functions for formulas pertaining to particular technique)

PART 1B USER REQUESTED DATA **

(This data DOES NOT include the Technique Penalty Equations (TPEs). They are generated PART 3)

QUESTION	VARIABLE ASSIGNMENT
1. How many primary input pins are used by the PCBs Operational Circuitry ?	v1
2. How many primary output pins ?	v2
3. How many test patterns are to be stored in the ROMs ?	v3
4. What is the test pattern application rate ?	v4
5. What is the estimated initialization rate ?	v5
6.	v6
7.	v7
8.	v8
9.	v9
10	v10

Figure 3-42

Sample Penalty Formula Generation WorkSheet For On-Board ROM Technique

PART 2 COMPONENT DETERMINATION EQUATIONS *****

(These formulas DO NOT include the Technique Penalty Equations (TPEs))

CIRCUIT TYPE	QUANTITY	FORMULA
ROM	n1	(v1/8) (v3/2048) + (v2/8) (v3/2048)
MULTIPLEXER	n2	v1/8
COMPARATOR	n3	v2/8
COUNTER	n4	v3/4
DECODER	n5	1 + (n4/3)
DELAY-CHIPS	n6	2
JK-FLIP-FLOP	n7	1
CONTROL-GATES	n8	2

PART 3 TECHNIQUE PENALTY EQUATIONS

(see Data Base of BIT Functions for formulas pertaining to penalty parameters (TPE's))

PENALTY PARAMETER	FORMULA
AREA	1.15 ((.375 n1) + (.87 n2) + (.375 n3) + (.243 n4) + (.375 n5) + (.375 n6) + (.243 n7) + (.243 n8))
POWER	(225 n1) + (350 n2) + (375 n3) + (455 n4) + (375 n5) + (200 n6) + (90 n7) + (110 n8)
WEIGHT	1.1 ((6.5 n1) + (7.5 n2) + (6.5 n3) + (2.0 n4) + (6.5 n5) + (6.0 n6) + (2 n7) + (2 n8))
TESTTIME	(v3) (v4) + v5
DELAY	N/A

Figure 3-42 (continued)

Sample C programs Generated

t01/formulas.c

```
# [ include statements ]
t01_form(area, weight, power, testtime, delay) {
            /* variable declarations */
            int v1, v2, v3, v4, v5, n[9];
            char *circuit[9];
            etc.....
            /* ask questions and read answers specified in figure 3-38 */
            /* use formulas to calculate penalties */
            /* print circuit names and formula results to  data/<profilename>—<t##> */

} /* end t01_form */
```

selection/penalty.c

```
# [ include statements ]
main ( ) {

        /* variable declarations */
        /* read technique numbers and name from  data/<profilename>/suitable */
        /* C switch statement from data/penalty-cases  placed here */
        /* calls t##_form( parameters ) */
        normalize ( parameters );
        weights ( parameters );
        rank_penalty ( parameters );
        /* print information to data/<profilename>/penalty-report */
} /* end */
normalize(parameters)
/* variable declarations */
{
        /* apply normalization formulas */
} /* end */
rank_penalty(parameters)
/* variable declarations */
{
        /* apply ranking formulas to obtain technique Figures Of Merit (FOM) */
} /* end */
```

selection/weighting-factors.c

```
# [ include statements ]
weights( parameters ) {

        /* variable declarations */

        /* read in weighting factors from profile and incorporate into normalized array */

} /* end */
```

Figure 3-43

alter the data file *data/penalty-cases* by adding in a subroutine call to the new C program. The data file is then incorporated into *selection/penalty.c*. If new penalties have been added to *data/template*, they are incorporated into *selection/weighting-factors.c*. The resulting C programs, *<t##>/formulas.c*, *selection/weighting-factors.c*, and *selection/penalty.c*, are then compiled and ready for use in generating the penalty report *data/<profilename>/penalty-report*.

(3) Outputs

The compiled C programs, *<t##>/formulas.c*, *selection/weighting-factors.c*, and *selection/penalty.c* (the main calling procedure), are all generated by the fix-penalty utility.

3.5 ADAPTATION REQUIREMENTS

CAD-BIT has been designed to be extensible, therefore, the whole data base is adaptable. This extensibility exists by virtue of the fact that BIT techniques can be added without limit (subject to memory storage limitations). Other factors which might limit extensibility may be freely added as line items in ASCII text files.

3.5.1 SYSTEM ENVIRONMENT

CAD-BIT is designed to run on various CAD systems. Certain operations, such as BIT Insertion, need to implement CAD graphics commands. For CAD-BIT to be extensible, the CAD graphics commands cannot be hardcoded into any executeable code. The data file *data/host-commands* is present on the CAD-BIT system to alleviate this problem. This file contains the syntax for various CAD systems for inserting parts into a PCB functional design. CAD-BIT can be adapted to an additional CAD system by adding the system's circuit insertion command syntax definition to the syntax file. The syntax definition is added in the first line of the file. A CAD Host Syntax file is shown in Figure 3-51.

The CAD-BIT graphics files, such as the tutorial figures and 3TIDs, are adaptable when translated via IGES on the CAD system being implemented.

3.5.2 SYSTEM PARAMETERS

The data used within the CAD-BIT system can change with the addition or deletion of BIT techniques and their corresponding BIT files (BIT circuit elements, BIT attributes, penalty parameters, etc). The following files give CAD-BIT its adaptability to system parameters : *data/default-parts*, *data/ppd*, *<t##>/circuitmenu*, *data/template*, *<t##>/teq-file*, *data/technique-list*, *<t##>/short-tut*, *<t##>/long-tut/<option>*, *help/<variable>-help*, and *<t##>/formulas.c*.

Additional technique circuits are added to the files *<t##>/circuitmenu*. Additional programs required for added techniques or circuits are added via the utilities and these programs become part of the catalogue structure. IGES files are used only to transfer

graphics files to different CAD systems. Line entries in the files *data/default-parts* and *data/ppd* need to be added only if new parts are added for the technique circuits. Additional penalty factors may be added by the addition of the penalty parameter in the PPL section of the file *data/template*.

Additional suitability attributes along with their respective default answers, allowable answers (SAA) and question text are added in the SAL section of the *data/template* file as well as in the BIT Technique files (*<t##>/teq-file*).

3.5.3 SYSTEM CAPACITIES

There are no capacity limitations built into the CAD-BIT system except that of disk storage space.

3.6 QUALITY FACTORS (REQUIREMENTS)

The quality factors for the CAD-BIT system are stated in the following subparagraphs.

3.6.1 CORRECTNESS REQUIREMENTS

The CAD-BIT system's objective is to help select the optimum BIT technique for a PCB functional design. To achieve this, numerous formulas are necessary. Of major importance is the quantity of BIT circuit elements inserted into the PCB. The formulas which calculate these numbers will be made accurate. Round off errors will be taken into account. If there is a fraction of a BIT circuit, that number will be rounded up to the next highest whole number.

3.6.2 RELIABILITY REQUIREMENTS

The CAD-BIT system will detect errors during its execution. Once detected, the user is informed of the name of the routine where the error occurs and any file names and parameters that have caused the error. If an error occurs in a data file, then the CAD-BIT system will output an error message. This message contains the data file and pointer to the line where the error or missing information is (i.e. technique name, suitability parameter, penalty parameter, etc.). After the error is detected, the user can alter the data file and rerun the procedure. For example, if an error occurs during the generation of a "DO" command in the BIT Insertion procedure, there is probably an entry missing in either *data/default-parts* or *<t##>/circuitmenu*. once the user alters these file(s), the BIT Insertion procedure can be rerun. If an error occurs in a C program, then the user runs the appropriate utility to fix the problem. For example, if prior to generating the estimated Penalty Report the user neglected to enter a parameter into *utilities/fix-penalty*, then the C routine it created will fail. An error message will be printed indicating the C program where the failure occurred and the parameter it failed on, if possible.

3.6.3 INTEGRITY REQUIREMENTS

The CAD-BIT system code shall restrict unauthorized write access to any of the the BIT data base files and code.

3.6.4 USABILITY REQUIREMENTS

Since the CAD-BIT system is a user-friendly package, little time and effort will be required for the engineer to understand the module. Through the use of unambiguous screens, menus, prompts, and on-line documentation, the engineer can walk through CAD-BIT with relatively little trouble. There is, however, no control over the CAD system operations.

3.6.5 MAINTAINABILITY REQUIREMENTS

If an error due to user input occurs during the generation of the User Design Profile, the user shall be informed of the error along with an explanation via the help facilities. If an error due to input from a data file occurs, an error message will be printed indicating the particular file and parameter causing the problem so the error can be corrected. When changes need to be made, they will be made to a minimun amount of items and accepted globally where applicable. These changes will either be made manually by the user or through the various utilities available.

3.6.6 PORTABILITY REQUIREMENTS

Since UNIX is written primarily in the C language and makes very few assumptions about the architecture of the computer, it has been transported to many different computer systems with a relatively small amount of effort. If a system supports both UNIX and C, most, if not all, of the CAD-BIT software shall run without change. CAD-BIT is structured so that no hard links to a particular system are required. It is generic in all areas.

3.6.7 INTEROPERABILITY REQUIREMENTS

It is of fundamental importance that CAD-BIT be able to interact with the operating systems on various CAD systems. For this reason, CAD-BIT shall be developed in C and UNIX C-Shell. Any system that supports UNIX should be able to support CAD-BIT.

3.7 COMPUTER SOFTWARE CONFIGURATION ITEM (CSCI) SUPPORT

The only facility required for the development, operation, and support of CAD-BIT is a CAD system which supports the UNIX operating system and the C programming language, as well as a host schematic graphics data extraction language. A designer who is an electrical circuit design engineer is required to operate this system.

3.8 TRACEABILITY

During a session of CAD-BIT, if there is an error, the error messages shall be structured to indicate the file where the error has occurred along with a brief explanation of why there is a problem. For example, in the Evaluation procedure, if there is an attribute missing on one of the circuit elements in the PCB, then when the information is extracted, the data file will be incomplete. When the procedure to generate the BIT Technique Penalty Report is run, it will generate an error which will specify the attribute missing in the file(s) so the user can easily identify and correct the problem.

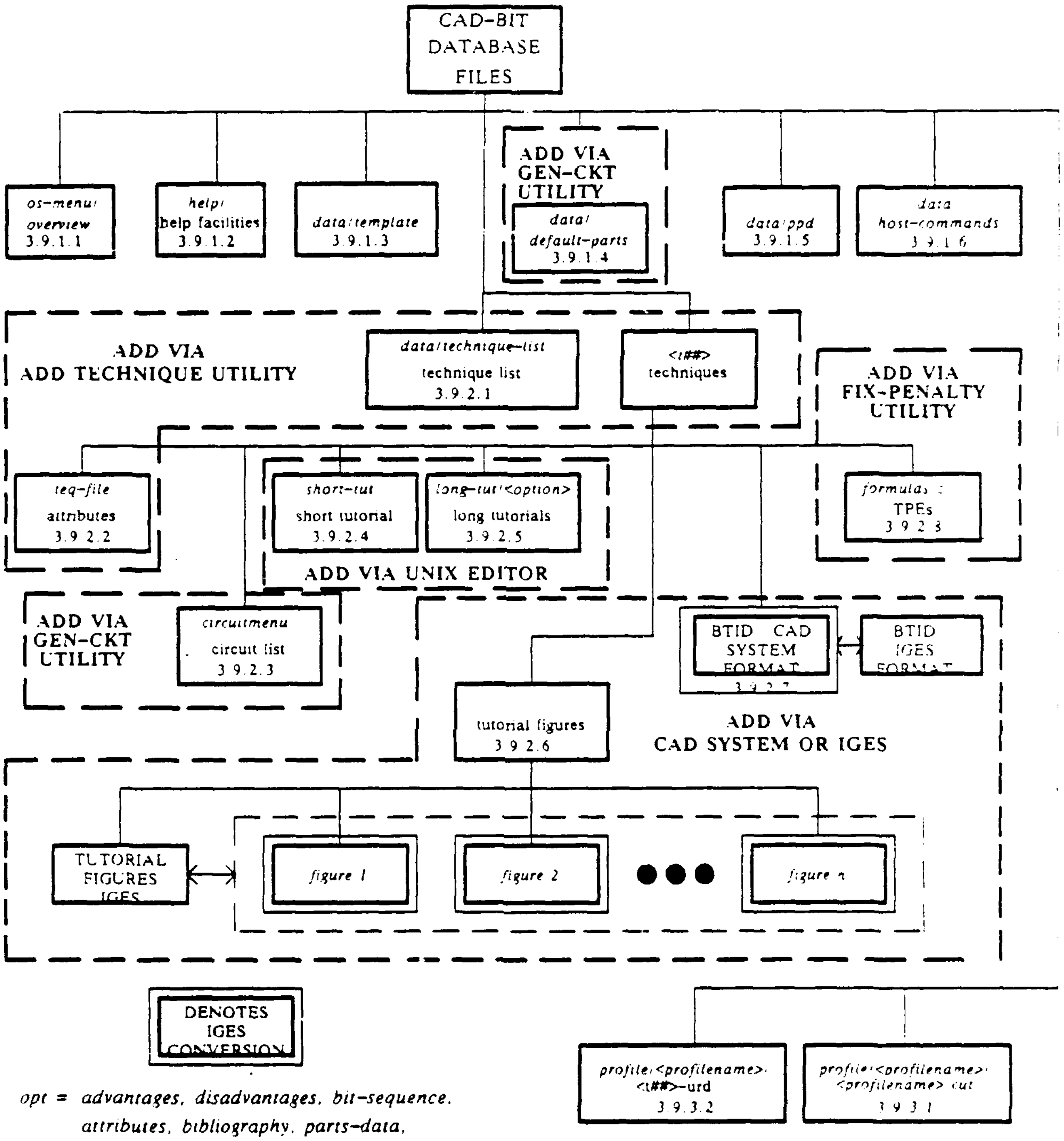
3.9 CAD-BIT DATA BASE FILES

The following subparagraphs define the contents of the CAD-BIT data base files. The particular data elements including text, figures, and formulas are contained in Volume II, Data Base of BIT Functions. Sample files are included in this volume. See Figure 3-44 for the position of the data elements in the CAD-BIT file structure. There are three divisions in the file structure. The first, system related files, contain non-technique dependent and non-design dependent data. The second, technique related files, are those files which are generated from the BIT Library Elements. They are all contained under the directory *<t##>*. If they need to be deleted, just delete the appropriate *<t##>* directory. The third and final division are the design related files. They are user generated but may exist on the CAD-BIT system after creation.

3.9.1 SYSTEM RELATED FILES

3.9.1.1 Overview

The overview gives the user a brief description of CAD-BIT. This description is contained in a standard UNIX text file whose name is *os-menu/overview*. The contents are shown in Figure 3-45.


CAD-BIT DATABASE FILES
os-menu/ overview 3.9.1.1
help/ help facilities 3.9.1.2
data/template 3.9.1.3
ADD VIA GEN-CKT UTILITY
data/ default-parts 3.9.1.4
data/ppd 3.9.1.5
data/ host-commands 3.9.1.6
ADD VIA ADD TECHNIQUE UTILITY
data/technique-list technique list 3.9.2.1
<t##> techniques
ADD VIA FIX-PENALTY UTILITY
teq-file attributes 3.9.2.2
short-tut short tutorial 3.9.2.4
long-tut/<option> long tutorials 3.9.2.5
ADD VIA UNIX EDITOR
formulas : TPEs 3.9.2.3
ADD VIA GEN-CKT UTILITY
circuitmenu circuit list 3.9.2.3
BTID CAD SYSTEM FORMAT 3.9.2.7
BTID IGES FORMAT
tutorial figures 3.9.2.6
ADD VIA CAD SYSTEM OR IGES
TUTORIAL FIGURES IGES
figure 1
figure 2
figure n
DENOTES IGES CONVERSION
profile/<profilename>/ <t##>-urd 3.9.3.2
profile/<profilename>/ <profilename> cut 3.9.3.1

opt = *advantages, disadvantages, bit-sequence, attributes, bibliography, parts-data,* and/or *default-design.*

Figure 3-44
CAD-BIT FILE STRUCTURE

os–menu/overview

The CAD–BIT module is made up of two different environments. There is a graphics CAD–BIT environment and an operating system CAD–BIT environment. The graphics CAD–BIT activities are controlled through the graphics CAD–BIT environment. The operating system CAD–BIT environment controls all CAD–BIT operations taking place in UNIX. You are currently in the CAD–BIT operating system environment.

The graphics CAD–BIT environment is used to perform all graphics operations. This menu will sometimes refer the user to the operating system menu for any CAD activities occurring in UNIX.

The Selection and Tutorial modules are accessed through the operating system CAD–BIT environment. The BIT Insertion and Evaluation modules are accessed through both the operating system and graphics environments. The menu items in each environment will prompt the user when it is necessary to change environments.

Figure 3–45
CAD–BIT OVERVIEW DESCRIPTION

3.9.1.2 Help

The purpose of the help facilities available with CAD-BIT is to give the user on-line documentation about the BIT technique attributes available and the weighting factors applied to the technique penalty calculations.

The help facilities are standard UNIX text files containing a brief description of the attributes/parameters being accessed in the profiles and BIT technique files. All of these help files are located under the directory *help* and have the structure */<variable>-help*, where *<variable>* represents all the attributes found in the PCB design profiles such as the type of circuitry involved, the technology used, etc. These are compared with the various BIT technique attribute files during suitability determination. Penalty parameters are also represented in the help facilities. They have the same file structure as the attribute help files. See Figure 3-46 for a sample help file.

To add a help file, change the current working directory to *help*. Open a file with the name */<variable>-help* using a standard editor command and type in the information about the attribute/parameter. Once complete, save and close the file. Figure 3-47 illustrates adding BIT suitability attributes and associated help files.

The following entries are used to generate the help files in the CAD-BIT data base. They give a brief explanation of the attributes which can be used to determine the suitability of various techniques. The following are suggested entries, some of which are not implemented. If a new technique is added which needs these atributes, they can be added at a later date.

A. USAGE

Usage determines which of the three main branches of the BIT Technique groups will be considered, i.e. Digital, Analog or Hybrid. The data base may be expanded to include other groupings such as RF or Optical.

Question - Is the Line Replaceable Module Circuit Under Test (CUT) Analog, Digital or Hybrid?

B. CONCURRENT

Under certain conditions, the operational circuitry on the CUT must be tested without interrupting its function. Certain BIT techniques allow the CUT to be tested concurrently with its operation.

```
THE ATTRIBUTE USAGE DETERMINES WHICH OF
THE THREE MAIN BRANCHES OF BIT TECHNIQUES
WILL BE CONSIDERED.
    "D" FOR DIGITAL,
    "A" FOR ANALOG,
    "H" FOR HYBRID.

QUESTION: IS THE LINE REPLACEABLE MODULE
          CIRCUIT UNDER TEST  DIGITAL,
            ANALOG, OR HYBRID ?



    ** ONLY FIRST CHARACTER IS USED

  ** UPPER OR LOWER CASE IS ACCEPTABLE
```

Figure 3-46

**SAMPLE HELP FILE
FOR PROFILE USAGE QUESTION**

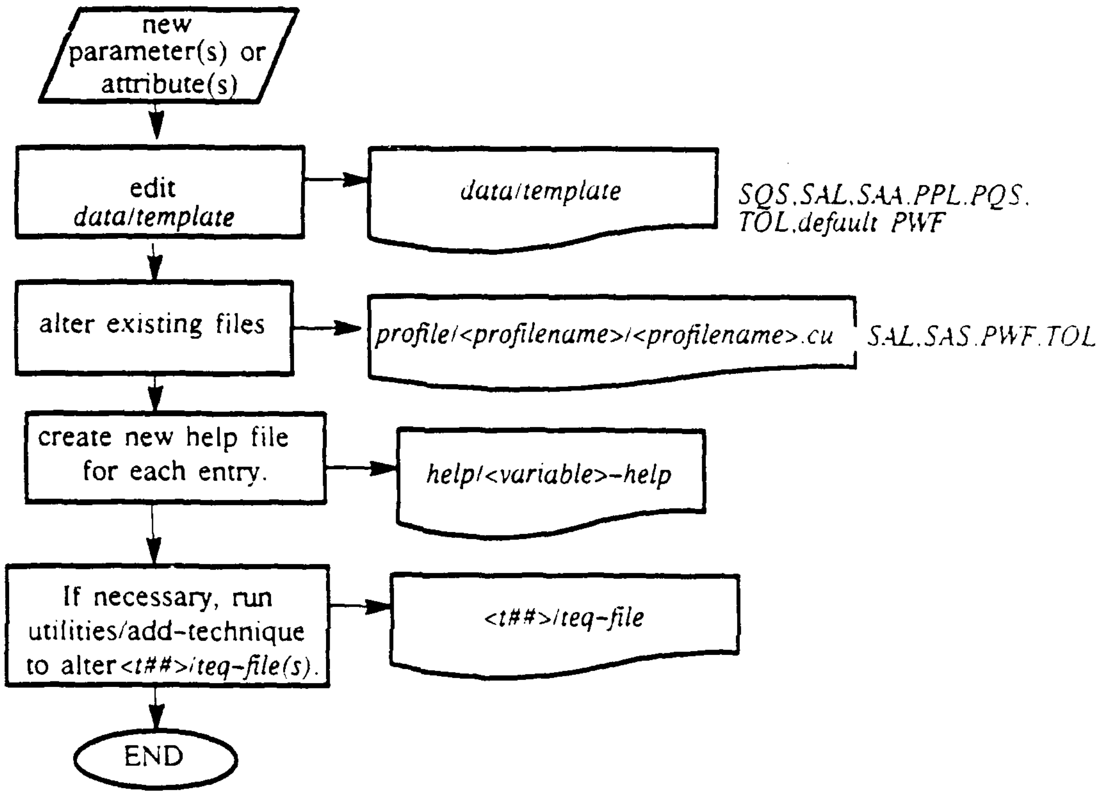

new parameter(s) or attribute(s)
edit data/template
data/template
SQS.SAL.SAA.PPL.PQS. TOL.default PWF
alter existing files
profile/<profilename>/<profilename>.cu
SAL.SAS.PWF.TOL
create new help file for each entry.
help/<variable>-help
If necessary, run utilities/add-technique to alter <t##>/teq-file(s).
<t##>/teq-file
END

Figure 3-47

ADDING NEW ATTRIBUTES

Question - Is the desired BIT technique required to be a concurrent one?

C. INTERNAL

Certain BIT techniques only require the addition of circuitry external to the CUT. Other techniques require access to the interior of the CUT. Internal refers to techniques that require access or addition of BIT hardware to the CUT interior.

Question - Will the designer accept hardware additions or modifications to the interior of his CUT design for the purpose of introducing BIT capability into the design?

D. MICROPROCESSOR

Many LRM's have a microprocessor as part of the CUT design. The designer may wish to utilize this intelligence to control or actually perform the BIT function.

Question - Is a microprocessor available in the CUT and does the designer wish to utilize it to conduct his BIT?

E. ICBIT

Many new Very High Speed Integrated Circuits (VHSIC) and Very Large Scale Integrated Circuits (VLSIC) (including Application Specific Integrated Circuits) have self contained BIT within the IC package. A technique is available which will utilize the individual Chip's BIT to perform a BIT on the LRM.

Question - Does the design contain a large amount of IC's with internal BIT?

F. MICRODIAGNOSTIC

Certain microprocessors have their instruction repertoire defined by data stored in a ROM external to the microprocessor (microprogrammed machine). The microdiagnostic BIT technique can utilize this architecture to implement an efficient BIT function.

Question - Is there a microprocessor that is a microprogrammed machine?

G. DIGWRAP

Certain LRM's contain complimentary (paired) digital input and output devices. If the LRM a! o contains a microprocessor, it can be used to check out this interface with the Digital Wraparound BIT Technique.

Question - Is there a microprocessor on board with complimentary digital I/O devices?

H. ANAWRAP

Certain LRM's (see question Digital, Analog or Hybrid) contain complimentary Analog I/O devices (A/D and D/A). If the LRM also contains a microprocessor, it can be used to check out the interface with a Hybrid Technique called Analog Wraparound.

Question - Is there a microprocessor on board with complimentary analog I/O devices?

I. BILBO

Certain circuit structures can be programmed to act as a source of BIT test stimuli, a recorder of BIT responses, or as an interval device to provide CUT internal BIT controllability and Observability functions. This circuit structure is called BILBO and can be conveniently incorporated in a package and distributed wherever required on an LRM layout.

Question - Does the designer wish to use a uniform circuit design to perform a variety of BIT functions (stimuli, response, internal observability & controllability)?

The following help entries pertain to the penalty parameters.

J. AREA

Area is the amount of PC Board real estate that the circuits take up. If minimizing the area is important, then give it a high percentage. The answer must be numeric with a range from 0 to 100.

Question - Area weighting factor?

K. WEIGHT

This is the weight of the actual BIT circuits. If minimizing the weight is important, then give it a high percentage. The answer must be numeric with a range from 0 to 100.

Question - Weight weighting factor?

L. TESTTIME

Test time is the amount of ime it takes to test the circuit. If minimizing test time is important, then give it a high percentage. The answer must be numeric with a range from 0 to 100.

Question - Test time weighting factor?

M. DELAY

Delay is the amount of time a functional signal is delayed as it propagates through BIT circuits. If minimizing delay time is important, then give it a high percentage. The answer must be numeric with a range from 0 to 100.

Question - Delay weighting factor?

N. POWER

If saving power is important, then give it a high percentage. The answer must be numeric with a range from 0 to 100.

Question - Power weighting factor?

3.9.1.3 Profile Template File

The profile/technique attributes are contained in a file used to generate both a user design profile and individual BIT technique attribute files. The profile contains the users' criteria for Built-In-Test capability for a particular PCB design.

This file, *data/template*, is a standard UNIX text file, Figure 3-48, containing the following four columns of information : the attribute or penalty parameter, the default value assigned to that attribute/parameter, the allowable answers for the attribute/parameter, and the question the user is to be prompted with to obtain an answer. The SAL, SAA, SQS, PQS, PPL, default PWF, and TOL are contained in this file.

To add an attribute/parameter, one must also add a default answer for the question, a list of allowable answers (first characters only), and a variable that the value will be assigned to. This variable is also [illegible] constructing the help file, section 3.9.1.2. When all four pieces of information have been established, change the working directory to *data* and, using the standard editor, edit the file */template*. Add the four columns of information into the appropriate section (SAL, PPL, or TOL). If you wish to delete an attribute/parameter, follow the same steps as above, however, delete the four columns of information mentioned above.

When questions and attributes/parameters are added to *data/template*, one may also decide to alter the existing profiles by either manually editing them or by re-running the Profile Generation section of the Selection procedure, section 3.4.5.1. See Figure 3-47 for a flow diagram describing this process. In certain instances (i.e. a BIT technique

SUITABILITY ATTRIBUTE LIST (SAL)	DEFAULT ANSWER	SUITABILITY ALLOWABLE ANSWERS (SAA)	SUITABILITY QUESTION SET (SQS)
USAGE	D	DAHR	Circuitry Involved?
CONCURRENT	Y	YN	Concurrent testing?
INTERNAL	N	YN	Internal Design?
MICROPROCESSOR	Y	YN	Microprocessor in circuit?

PENALTY PARAMETER LIST (PPL)	PENALTY WEIGHTING FACTORS		PENALTY QUESTION SET (PQS)
AREA	10	#	Area weighting factor?
POWER	10	#	Power weighting factor?
WEIGHT	10	#	Weight factor?
DELAY	10	#	Delay weighting factor?
TESTTIME	10	#	Testtime weighting factor?

TECHNIQUE OVERRIDE LIST (TOL)	
EXCLUDE	Forced exclusion?
INCLUDE	Forced inclusion?

KEY:

# INDICATES THAT NUMERICAL ANSWER (POSITIVE INTEGER IS REQUIRED).

USAGE ATTRIBUTE QUESTION REQUIRES D (DIGITAL), A (ANALOG), H (HYBRID), or R (RF) ANSWER.

CONCURRENT, INTERNAL, AND MICROPROCESSOR QUESTIONS REQUIRE Y(YES) OR N(NO) ANSWER.

THIS AND ADDITIONAL INFORMATION IS INCLUDED IN THE HELP FILES. SEE SECTION 3.9.1.2.

Figure 3-48
TEMPLATE FILE

attribute being added or deleted and used to determine suitability), one must also alter the existing BIT Technique files, see section 3.4.8.1 and Figure 3-54.

3.9.1.4 Default Parts File

The file *data/default-parts*, a standard UNIX text file, contains information about the default parts which are inserted into the PCBs. This information is divided into three columns containing the circuit name(s), the library name(s), and the fault part number(s), Figure 3-49.

This file is used by the BIT Insertion section of CAD-BIT, section 3.4.6.1. To alter the file *data/default-parts*, run *utilities/gen-circuit*, described in section 3.4.8.2.

data/default-parts

CIRCUIT	**DEF. P/N**	**LIBR. NAME**
COMPARATOR	LM319N	CADBIT.LM319N
OP-AMP	MC1558	CADBIT.MC1558
●	●	●
●	●	●
●	●	●
JK-FLIP-FLOP	74H103	CADBIT.74H103
ROM	TBP38516	CADBIT.TBP38516

Figure 3-49

DEFAULT PARTS FILE

3.9.1.5 Parts Penalty Data

The file *data/ppd* contains part information such as part numbers, area, power, etc. This information is used during the evaluation of the BIT circuits added to the PCB. This file is a standard UNIX text file containing information on the penalties associated with each part number in the file. Figure 3-50 describes the file's contents. To alter it, use the standard editor and fill in the appropriate columns of information. See Volume II, Data Base of BIT Functions, for the Parts Penalty Data.

PART NO.	AREA	POWER		WEIGHT
LM319N	.21	.05		0.43
MC1558	.24	.09	●●●	0.43
TBP38516	.24	.09		0.43
●	●	●		●
●	●	●		●
●	●	●		●
74H103	.25	.03		0.43

Figure 3-50
PARTS PENALTY DATA

3.9.1.6 CAD Host Syntax

The file *data/host-commands* contains a list of various CAD system insertion commands. The first entry is the one used in the BIT Insertion section of CAD-BIT. This file is a standard UNIX text file containing three columns of information : the CAD system name, the CAD command syntax, and any additional text or punctuation which is part of the command. This file is shown in Figure 3-51. The command structure shown is followed by the CAD library figure name in the default parts file. To add an entry to this file, use the standard editor and fill in the appropriate columns of information. The line corresponding to the implemented CAD system must be the first line of the file.

CAD SYSTEM	SYNTAX	TEXT
CV	Insert Nfig	:
MENTOR	Activate Component	
DAISY	Component	
CV-SCHEM.	Get Symbol	

Figure 3-51

CAD HOST SYNTAX FILE

3.9.2 TECHNIQUE RELATED FILES

3.9.2.1 Technique List

The file *data/technique-list*, Figure 3-52, contains a list of all the techniques for user reference as well as for menu display. This file is a standard UNIX text file containing two columns of information: the technique number and technique name. To add/delete, alter a technique in the list, use the *utilities/add-technique*, section 3.4.8.1. Figure 3-53 illustrates the process of adding a technique.

3.9.2.2 BIT Technique Attributes

The file(s) *<t##>/teq-file* are the individual BIT technique files containing the attributes and the corresponding values required for them to be considered suitable for a particular PCB design. The BIT technique files are standard UNIX text files containing the following two columns of information : the technique attribute(s), and the answer(s) assigned to them, Figure 3-54. This data is found in Volume II, Data Base of BIT Functions, for each technique. Use the standard editor to add in a technique attribute and answer(s) into the BIT Technique file(s) or add to the file *data/template* first and then run the utility */utilities/*

data/technique–list

t01	On–Board ROM
t02	PRPG/MISR
t03	Microprocessor BIT
t04	OBIVCB
t05	Microdiagnostics
t06	Error Detection and Correction Codes
t07	Digital Wraparound
t08	BILBO
t09	Scan
t10	Comparator
t11	Redundancy
t12	Voltage Summing
t13	Analog Wraparound

This file contains a listing of all the available techniques and their technique numbers.

Figure 3–52
TECHNIQUE LIST

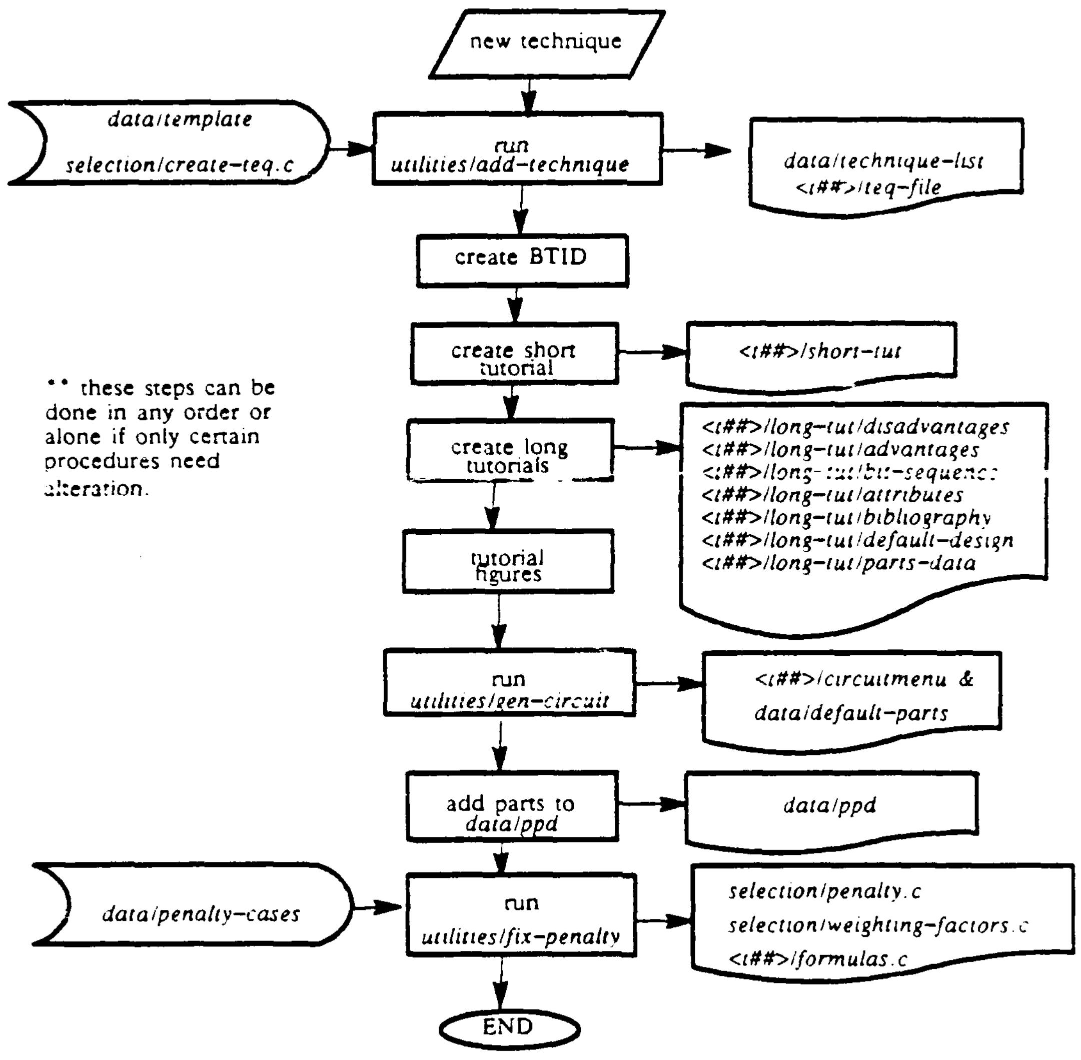


Figure 3–53

DDING A NEW TECHNIQUE

add-technique, section 3.4.8.1, to edit/overwrite the existing BIT technique file. When an attribute is added or deleted, however, it must be added or deleted in the file *data/template*.

IDENTIFICATION SECTION	SUITABILITY ANSWER SET (SAS)
ID	t01
NAME	On-Board ROM

SUITABILITY ATTRIBUTE LIST (SAL)	SUITABILITY ANSWER SET (SAS)
USAGE	Digital
CONCURRENT	No
INTERNAL	No
MICROPROCESSOR	No

Figure 3-54
BIT TECHNIQUE ATTRIBUTE FILE

3.9.2.3 Technique Circuit List

The file *<t##>/circuitmenu* contains a list of all the circuits and their default part numbers for a particular technique. These files are used as both menus and as inputs along with *data/default-parts* into the BIT Insertion portion of CAD-BIT. The circuit lists are standard UNIX text files containing the circuit names and default part numbers. Figure 3-55 illustrates this file. This data is found in Volume II for each technique. Run the *utilities/gen-circuit* procedure, section 3.4.8.2, to alter this file.

3.9.2.4 Short Tutorial Text

The Short tutorials, *<t##>/short-tut*, are standard UNIX text files containing a brief description of the BIT techniques used in the CAD-BIT system. The data for these files is found in Volume II for each technique supplied. See Appendix-A, Short tutorial category, page 1 for a sample Short tutorial. To add a Short tutorial, use the standard editor command. If a technique directory does not already exist, make a directory *<t##>*.

Under this directory, create the file *short-tut* and enter the information explaining the technique.

<t##>/circuitmenu

CIRCUIT	DEFAULT P/N
1. COMPARATOR	LM319N
2. OP-AMP	MC1558
3. JK-FLIP-FLOP	74H103
●	●
●	●
●	●
#. ROM	TBP38516

Figure 3-55

TECHNIQUE CIRCUIT LIST

3.9.2.5 Long Tutorial Text

The Long tutorials, *<t##>/long-tut/<option>* (where *option* represents advantages, disadvantages, bit-sequence, bibliography, attributes, parts-data, and/or default-design) are standard UNIX text files containing detailed descriptions of the BIT techniques used in the CAD-BIT system. This data is found in Volume II for each technique supplied. To generate a Long tutorial, if a technique directory does not already exist, create a directory <t##>/long-tut. Using the standard editor, edit the file *<option>*, where *<option>* represents advantages, disadvantages, bit-sequence, bibliography, attributes, parts-data, and or default-design. See Appendix-A, Long tutorial category, pages 1-14 for sample contents.

3.9.2.6 Tutorial Figures

The tutorial figures, *<cadsys>/cadbit/<t##>/<fign>* (where *<cadsys>* is the host CAD system graphics directory) include all figures which aid in the selection, insertion, and interconnection processes. They can be viewed in the CAD host graphics environment to be used as guides along with the tutorials. Various figures in Appendix-A can be used as tutorial figures. IGES equivalent figures are represented as *<t##>/IGES/<fign>* and are translat· into the host CAD graphics format via IGES.

3.9.2.7 BIT Technique Insertion Diagrams (BTID)

Inserted into the PCB prior to the insertion of the actual BIT circuitry, the BTIDs act as guides and check lists for the user to follow, Figure 3-56. They contain boxes representing the circuits involved which are arranged to guide the user on how to do the actual insertions. As each piece is inserted, its box in the BTID is checked off to mark the circuit as being completed, Figure 3-57. The BTID is a particular tutorial file that must be included for all techniques and has the same location as the tutorial figures in the *<cadsys>* directory.

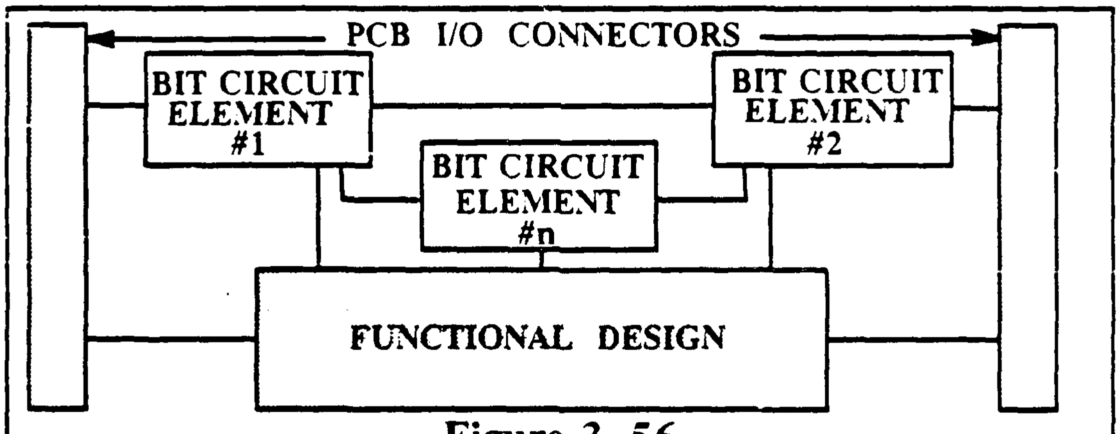
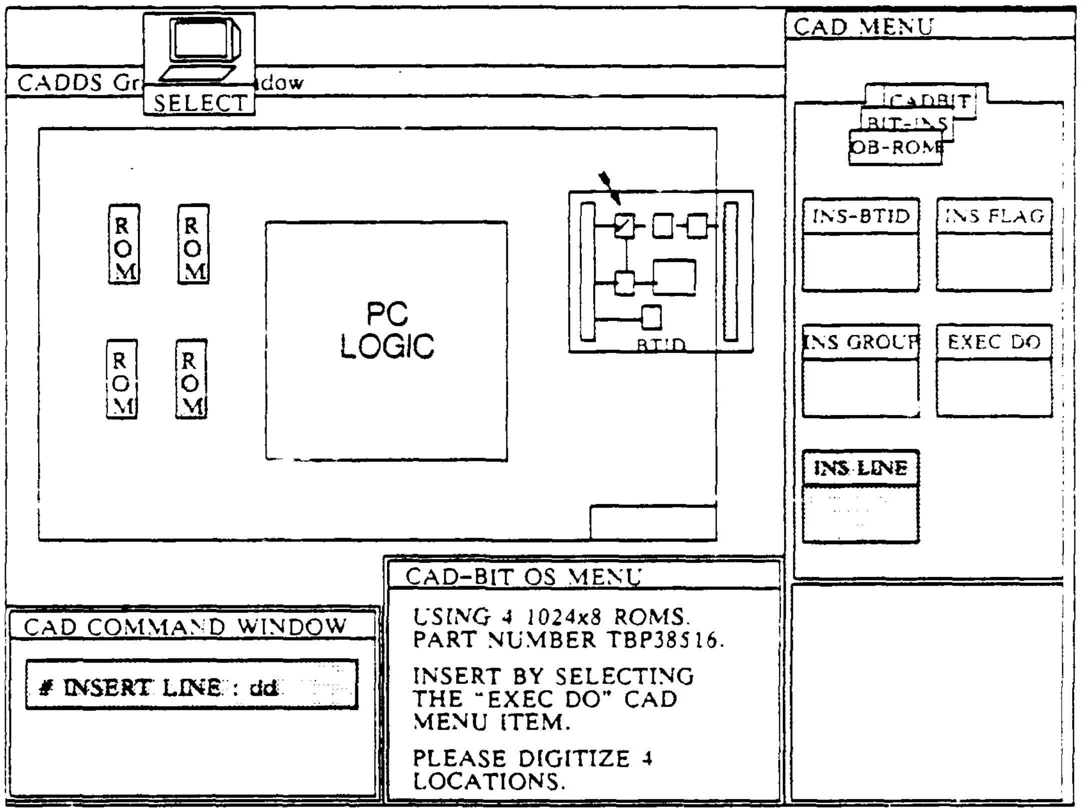


Figure 3-56
BIT TECHNIQUE INSERTION DIAGRAM (BTID)



INSERT LINE : dd

USING 4 1024x8 ROMS.
PART NUMBER TBP38516.

INSERT BY SELECTING THE "EXEC DO" CAD MENU ITEM.

PLEASE DIGITIZE 4 LOCATIONS.

Figure 3-57

BTID ROM CHECKOFF: LINE THROUGH ROM BOX IN BTID (AT ARROW)

3.9.2.8 Technique Penalty Equations

The Technique Penalty Equations (TPEs) are used during the Penalty Calculation portion of the Selection procedure. The equations are incorporated into the C programs *<t##>/formulas.c* and are accessed when *selection/penalty.c* calls the appropriate C programs for a technique. The source data for this may be found in Volume II, Data Base of BIT Functions, in the following sections for each technique :

- User Requested Data (URD)
- Component Determination Equations (CDEs)
- Technique Penalty Equations (TPEs)

When added via the *utilities/fix-penalty* utility, this "data" will reside in a C program for each technique. To add equations for a new technique, run *utilities/fix-penalty* to create a new C program, *<t##>/formulas.c*, for the technique. The data file, *data/penalty-cases*, is also altered by this utility to contain a call to the new C program. Figure 3-58

data/penalty-cases

```
case (1) :

          ierr = t01_form(param(i,1),param(i,2),....param(i,j));

case (2) :

          ierr = t02_form(param(i,1),param(i,2),....param(i,j));

case (n) :

          ierr = t0n_form(param(i,1),param(i,2),....param(i,j));

default :

          printf("No such BIT Technique number t%d...\n",ans);
```

Figure 3-58
SAMPLE C SWITCH BLOCK

3.9.3 DESIGN RELATED FILES

3.9.3.1 User Design Profiles

The profiles are divided into four sections as follows.

The identification section contains the name of the profile, the name of the user, and any nomenclature entered by the user which has no bearing on the Selection process. It is the first section the file(s) of *profile/<profilename>/<profilename>.cut*, Figure 3-59.

The SAS contains the user generated answers for each attribute in the profile. This list can be found in the second section of the file(s) *profile/<profilename>/<profilename>.cut*, Figure 3-59, and has two columns of information, the SAL and the SAS.

The PWF contains the user generated penalty weighting factors to be used in the generation of the penalty report for a particular profile. This list can be found in the third section of file(s) *profile/<profilename>/<profilename>.cut*, Figure 3-59, and has two columns of information, the PPL and PWF.

The TOL contains the user generated technique overrides for a particular profile. This list is used to eliminate certain BIT Techniques from consideration and can be found in the file(s) *profile/<profilename>/<profilename>.cut*, Figure 3-59. It is the fourth section of *profile/<profilename>/<profilename>.cut* and has two columns of information, the TOL variable and the technique(s) to be overridden.

3.9.3.2 User Requested Data Files

These files, *profile/<profilename>/<t##>-urd*, Figure 3-60, are created during the Penalty calculation section of the Selection procedure for each suitable technique. They contain all the user responses to the User Requested Data per technique as well as the variables that the answers are assigned to. *If these files exist during the penalty calculation section,* the user is prompted as to whether or not the old answers are to be used. If so, the data is taken from the appropriate file instead of from the user. If the user does input new data, the old file is overwritten.

IDENTIFICATION SECTION	SUITABILITY ANSWER SET (SAS)
USER	JOHN ENGINEER
CIRCUIT ID	ABC-123
DESCRIPTION	SIGNAL PROCESSOR-1

SUITABILITY ATTRIBUTE LIST (SAL)	SUITABILITY ANSWER SET (SAS)
USAGE	D
CONCURRENT	Y
INTERNAL	N
MICROPROCESSOR	Y

PENALTY PARAMETER LIST (PPL)	PENALTY WEIGHTING FACTORS
AREA	40
POWER	10
WEIGHT	10
DELAY	5
TESTTIME	10

TECHNIQUE OVERRIDE LIST (TOL)	
EXCLUDE	T05
INCLUDE	T01

Figure 3-59
SAMPLE USER DESIGN PROFILE: CBM GENERATED

profile/<profilename>/<t##>–urd

DATA	VARIABLE
8	v1
20	v2
2900	v3
5	v4
●	●
●	●
●	●
3	vn

Figure 3–60

USER REQUESTED DATA FILE

4.0 QUALIFICATION REQUIREMENTS

This section specifies the methods, techniques, tools, and acceptance tolerance limits necessary to ensure whether the CAD-BIT system satisfies the requirements of section 3.

4.1 DEMONSTRATION

An operational demonstration shall be developed to ensure that each of the requirements for the CAD-BIT system have been satisfied. The following areas will be covered in the demonstration :

- Overview
- Short tutorial
- Long tutorial
- Selection
  a. Profile generation
  b. Suitability determination
  c. Penalty calculations
- BIT Insertion
- BIT Evaluation
- Extensibility (utilities

4.2 ANALYSIS

The equations and diagrams used in the CAD-BIT system will be reviewed to verify that they are compliant with the standards specified. Such areas will include the calculations for the estimated Penalty Report, the Tutorial figures and BTIDs, etc.

4.3 TEST

There will be bench marks to demonstrate different activities and options available with the CAD-BIT system such as testing for the presence of help files in the Profile generation section.

4.4 INSPECTION

There will be a visual examination of the screen displays to verify their compatibility with the host system in the CAD and operating system environments.

5.0 PREPARATION FOR DELIVERY

The CAD-BIT developer shall supply a C-Shell script which will place all the CAD-BIT files in the necessary directories, and, if necessary, compile and link any C programs used in CAD-BIT. This C-Shell script will be located on a 1/2 inch magnetic tape separate from the CAD-BIT source code. This tape will be loaded onto the system and the C-Shell script executed. The C-Shell script will load the second tape containing the CAD-BIT libraries and software onto the system and make it operational.

6.0 INITIALIZATION & INSTALLATION

Figure 6-1 briefly illustrates all the steps necessary for the initialization of the CAD host system for CAD-BIT and the installation of CAD-BIT libraries and software. A more detailed explanation is located below.

6.1 CAD HOST SYSTEM ENVIRONMENT

Prior to loading CAD-BIT onto the CAD host system there must exist the following :

- 1/2 inch magnetic tape drive.
- UNIX based operating system.
- Multi-window capability.
- C compiler and associated libraries.
- CAD application software
- CAD extraction procedure.
- CAD graphics menu capability.
- Initial Graphics Exchange Specification (IGES).

If all of the above items exist, then the user must develop both a CAD-BIT Evaluation extraction program (see Section 3.4.7.1) and a CAD-BIT graphics environment menu.

6.2 CAD-BIT INSTALLATION

Two tapes are used for the installation of CAD-BIT software and libraries onto the CAD host system. The first tape contains two files. The first is a C-Shell script which creates a specified CAD-BIT originating directory and sets up the necessary sub-directories : */profile, /data, /<t##>, /<t##>/long-tut, /<t##>/insert, /utilities, /os-menu, /selection*, and */help*, and */eval*. Permissions are placed upon these directories to allow only certains types of access: read, write, and/or execute. Once complete, this C-Shell script will read the CAD-BIT software and libraries from the second tape. The second file of the first tape only needs to be used if the CAD-BIT source code has to be re-compiled. This file is a C-Shell script that loads and links all resident CAD-BIT C programs.

6.3 IGES CONVERSION

Once the CAD-BIT libraries are loaded onto the CAD host system, the tutorial figures and BTIDs must be translated from IGES neutral format into the CAD system format.

CAD-BIT INSTALLATION FLOW DIAGRAM

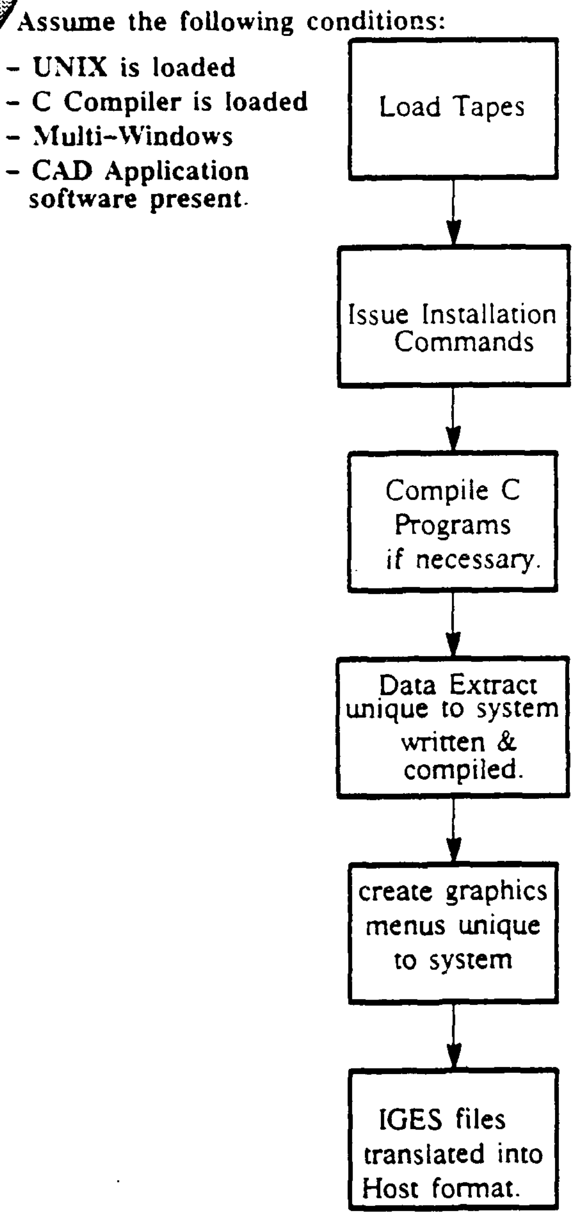

Assume the following conditions:
- UNIX is loaded
- C Compiler is loaded
- Multi-Windows
- CAD Application software present.
Load Tapes
Issue Installation Commands
Compile C Programs if necessary.
Data Extract unique to system written & compiled.
create graphics menus unique to system
IGES files translated into Host format.

Figure 6-1

7.0 NOTES

This section contains general information that may aid in understanding this specification (e.g., background information, glossary, formula derivations, definitions). Included is an alphabetical listing of all acronyms and abbreviations and their meanings.

7.1 DEFINITIONS

VARIABLE	DEFINITION
CDE	Component Determination Equations are the equations used during the Penalty calculation section of the Selection procedure which determine the quantity of the components to be inserted into the PCB schematic.
PPL	Penalty Parameter List contains a listing of all penalty variables used for the generation of the estimated Penalty Report.
PQS	Penalty Question Set contains a listing of all questions corresponding to the PPL.
PROFILE	This is an environment variable which represents the full name of the most recently used User Design Profile.
PWF	Penalty Weighting Factors contains a listing of the weights the user assigns to the PPL variables.
SAA	Suitability Allowable Answers contains a set of allowable characters which represent allowable answers for each SAL.
SAL	Suitability Attribute List contains a listing of the suitability variables used in the selection of suitable techniques.
SAS	Suitability Answer Set contains a listing of the answers the user assigns to the SAL variables.
SQS	Suitability Question Set contains a listing of all questions corresponding to the SAL.
TOL	Technique Override List contains a listing of all the techniques the user specifies to either be forcibly included or excluded.

VARIABLE	DEFINITION
TPE	Technique Penalty Equations are the equations which generate the BIT technique penalties during the Penalty calculation section of the Selection procedure.
TPP	Technique Penalty Parameters are the penalty parameters arrived at for each technique in the Penalty Calculation section of the Selection process. These parameters are those which have not had the PWFs applied to them.
URD	User Requested Data is the data which is requested from the user during the Penalty calculation section of the Selection procedure.

7.2 GLOSSARY OF TERMS

VARIABLE	MEANING
A/D, D/A	Analog/Digital, Digital/Analog.
BILBO	Built-In Logic Block Observer.
BIT	Built-In-Test.
CAD	Computer Aided Design.
CLIN	Contract Line Item Number.
CAD-BIT	Computer Aided Design for Built-In-Test.
CDE	Component Determination Equation.
CSCI	Computer Software Configuration Item.
CUT	Circuit Under Test.
CV	Computervision.
ELIN	Exhibit Line Item Number.
GASD	Grumman Aircraft Systems Division.

VARIABLE	MEANING
IC	Integrated Circuit.
ICBIT	Integrated Circuit BIT
ID	Identification.
I/O	Input/Output.
IGES	Initial Graphics Exchange Specification.
LRM	Line Replaceable Module.
MCBW	Main Cad-BIT Window.
N/A	Not Applicable.
OBIVCB	On-Board Integration of VLSI Chip Bit.
OS	Operating System.
PCB	Printed Circuit Board.
PPL	Penalty Parameter List.
PQS	Penalty Question Set.
PRPG/MISR	Pseudo Random Pattern Generator / Multiple Input Shift Register.
PWF	Penalty Weighting Factors.
RADC	Rome Air Development Center.
RFP	Request For Proposal.
SAA	Suitability Allowable Answers.
SAL	Suitability Attribute List.
SAS	Suitability Answer Set.
SOW	Statement Of Work.

SQS	Suitability Question Set.
VARIABLE	**MEANING**
----------	--------
SRS	Software Requirements Specification.
TOL	Technique Override List.
TPE	Technique Penalty Equations.
TPP	Technique Penalty Parameters.
URD	User Requested Data.
VHSIC	Very High Speed Integrated Circuits.
VLSI	Very Large Scale Integration.
VLSIC	Very Large Scale Integrated Circuits.
#	denotes a user supplied number or PWF digit.
##	denotes a technique number.
@	denotes standard UNIX command.
%	denotes host system UNIX prompt.

APPENDIX A
BIT LIBRARY EXAMPLE

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM	
CATEGORY: SHORT TUTORIAL	PAGE 1 of 4
SUBCATEGORY: DESCRIPTION OF BIT TECHNIQUE	

DATA TYPE: TEXT [✓] LIST ☐ TABLE ☐ GRAPHIC ☐ EQUATIONS ☐

DATA:

SHORT TUTORIAL
FOR
ON-BOARD ROM

On-Board Read Only Memory (ROM) Self Test is non-concurrent, mostly hardware and firmware, Built-In-Test (BIT) technique which consists of applying test patterns that are stored in an on board ROM to a Circuit Under Test (CUT) and then comparing the CUT's response to what is expected, resulting in a go - no/go output signal. Although the number of test patterns required to exhaustively test a function is proportional to the cube of the number of gates, this technique has some potential in that each test pattern can be individually and selectively determined, thereby, maximizing the percentage of fault detection to the test pattern ratio.

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM	
CATEGORY: SHORT TUTORIAL	PAGE 2 of 4
SUBCATEGORY: 1. LEVEL I BLOCK DIAGRAM. 2. BIT SEQUENCE FLOW CHART.	

DATA TYPE: TEXT ☐ LIST ☐ TABLE ☐ GRAPHIC [✓ ☒] EQUATIONS ☐

DATA:

SUBCATEGORY 1: SEE FIGURE 1

SUBCATEGORY 2: SEE FIGURE 2

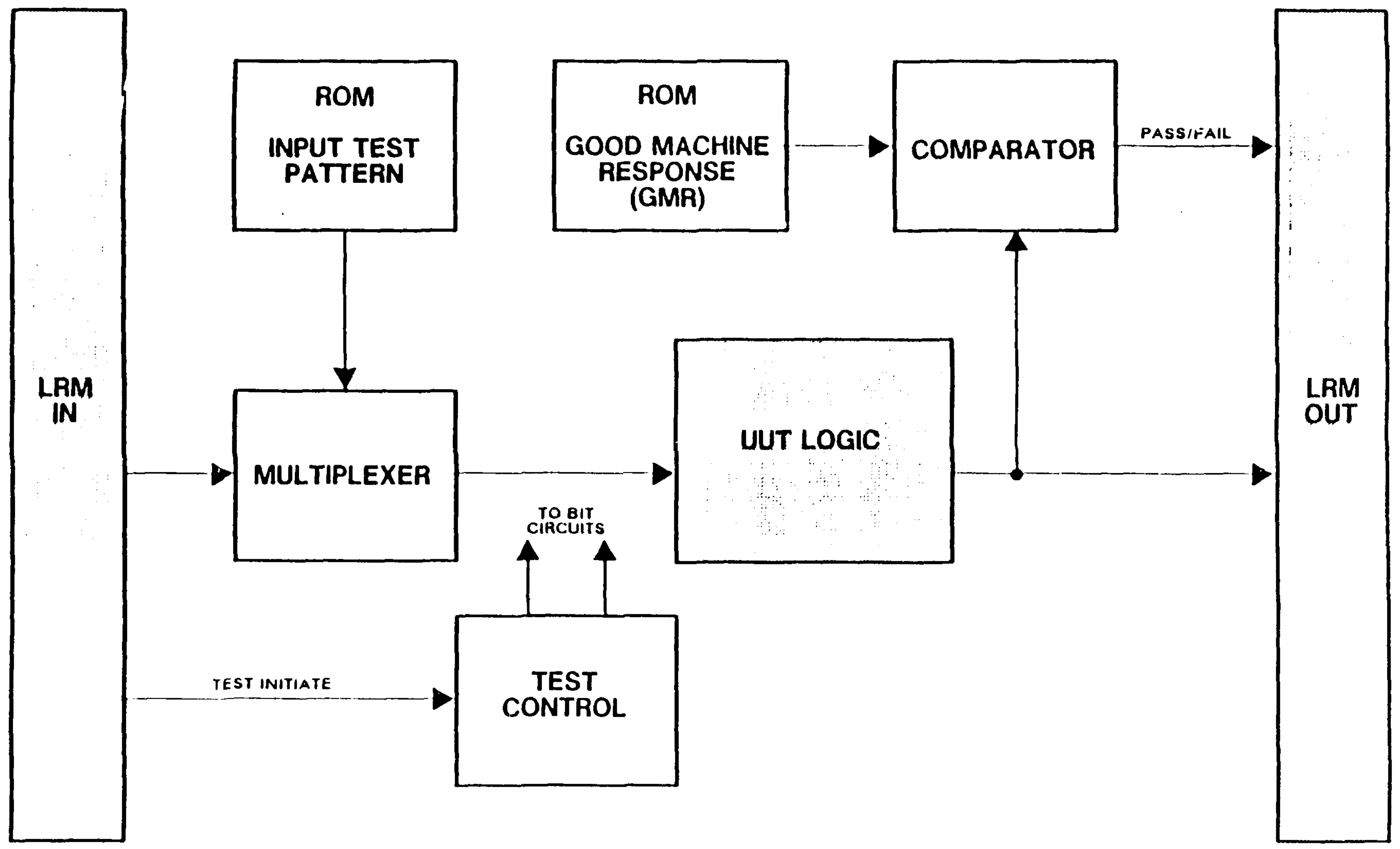


FIGURE 1 – LEVEL I BLOCK DIAGRAM FOR ON-BOARD ROM

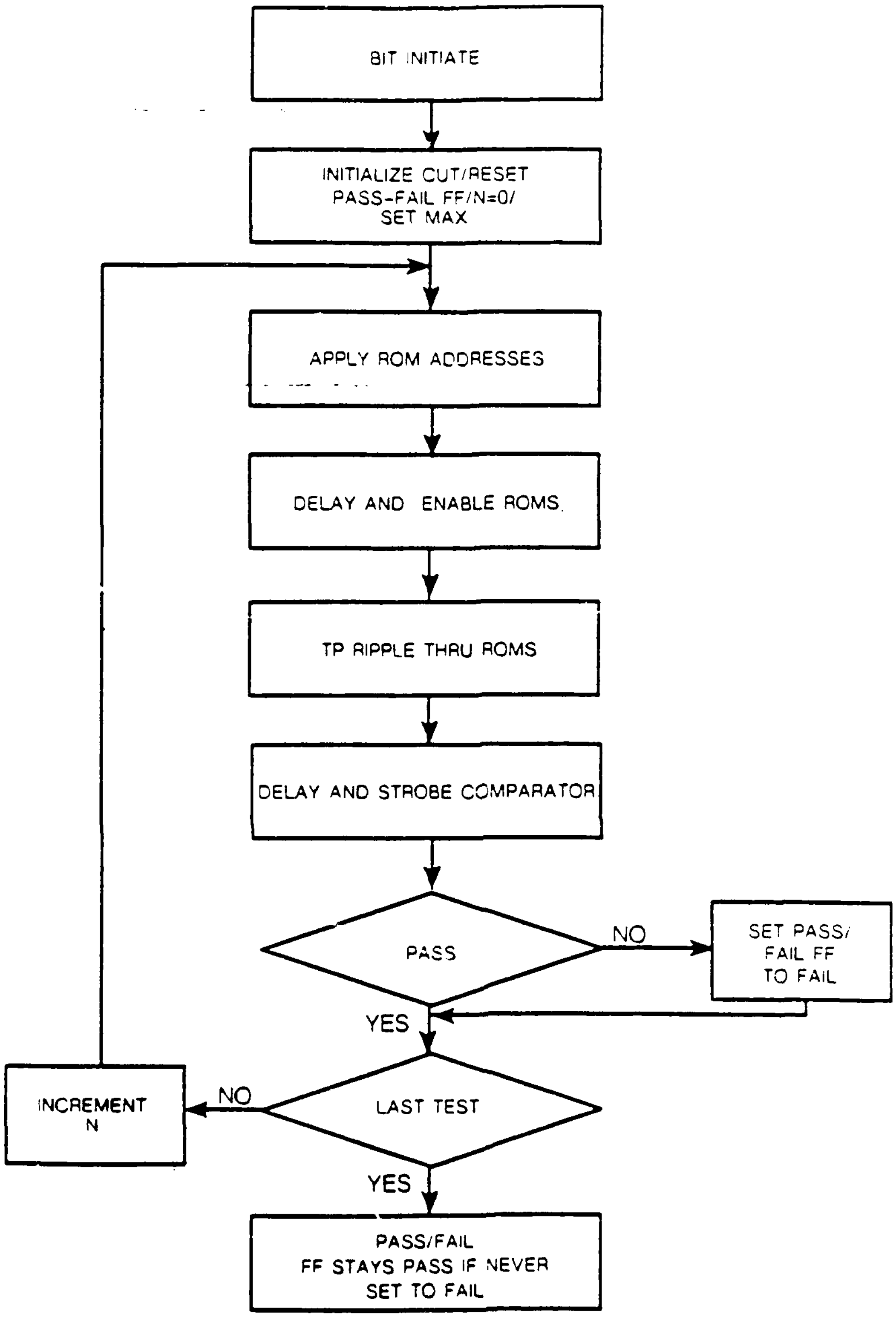

BIT INITIATE
INITIALIZE CUT/RESET PASS-FAIL FF/N=0/ SET MAX
APPLY ROM ADDRESSES
DELAY AND ENABLE ROMS
TP RIPPLE THRU ROMS
DELAY AND STROBE COMPARATOR
PASS
NO
SET PASS/ FAIL FF TO FAIL
YES
LAST TEST
NO
INCREMENT N
YES
PASS/FAIL FF STAYS PASS IF NEVER SET TO FAIL

FIGURE 2 – BIT SEQUENCE FLOW CHART FOR ON-BOARD ROM

PAGE 4 of 4

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM	
CATEGORY: LONG TUTORIAL	PAGE 1 of 14
SUBCATEGORY: BIT SEQUENCE FLOW CHART DESCRIPTION	

DATA TYPE: TEXT ☐ LIST ☒ TABLE ☐ GRAPHIC ☐ EQUATIONS ☐

DATA:

BIT SEQUENCE FLOW CHART DESCRIPTION FOR ON-BOARD ROM

1. A positive pulse "Test Initiate" signal is input to test control logic to begin test
2. The test begins as follows:
   - "BIT Mode" signal from control logic to multiplexer is activated
   - Normal inputs to CUT multiplexed out
   - Test Patterns (TP) from TP ROM input to CUT enable
   - All resettable logic of CUT reset
   - ROM address counter in control logic reset to zero
   - "Pass/Fail" Flip-Flop (FF) in comparator logic block reset to "Pass"
3. The system clock, while in BIT mode, increments the control logic counter which addresses the TP & Good Machine Response (GMR) ROMs simultaneously.
4. After a delay sufficient to fully establish the addressing in the step above, both the TP & GMR ROMs are enabled.
5. The TP ripples through the CUT. To gain control of the CUT clock, each TP will have both high and a low on the clock line which may come from the TP ROM.
6. After enough delay for a good machine to establish a GMR at the CUT's outputs, the comparator is enabled.
7. A good machine at this time will have the GMR pattern identically compare with the CUT outputs. If not, the Pass/Fail FF will be set to "Fail" and will remain "Fail" until BIT is re-initiated.
8. If the address to the ROMs is the last address, then "End Of Test" control logic signal goes low. The moment the enable comparator signal goes HI during this last TP sequence, the BIT mode FF is reset and the system is out of BIT mode. The Pass/Fail FF will remain set to "Pass" if during the test it was never set to "Fail".
9. If not the last ROM address, go back to step 3.

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM	
CATEGORY: LONG TUTORIAL	PAGE 2 of 14
SUBCATEGORY: BIT TECHNIQUE ADVANTAGES	

DATA TYPE: TEXT ☐ LIST ☒ TABLE ☐ GRAPHIC ☐ EQUATIONS ☐

DATA:

ON-BOARD ROM BIT
ADVANTAGES

1. An understanding of the CUT can lead to a substantial percentage of fault detected with a few, determined test patterns.

2. A CUT with much sequential logic requires specific "Pairs" of test patterns applied in sequence. Although, this presents a problem with Random Test Pattern Application, storing the test patterns in ROM so that they indeed do occur in pairs is done without difficulty with the On-Board ROM Method.

3. On-Board ROM Test Generation becomes competitive when compared to random pattern generation as the number of CUT inputs become large and/or number of patterns required becomes small. This is best understood by considering that the total number of binary patterns possible for a CUT with n inputs is 2^n. If n=16; $2^n = 65,536$. If n=20; $2^n = 1,048,576$. If n=24; $2^n = 16,777,216$. Consider a hypothetical 24 input CUT that can be adequately tested with 2,000 deterministic patterns. Most of the Test Pattern Generator (TPG) hardware required using On-Board ROM Method are cascaded, 2K by 8 ROMs as compared to 3 cascaded 8-Bit shift registers plus 2 Quad Exclusive Or Packages. But the real savings is test time. To be absolutely sure of providing all 2,000 test patterns one must cycle through 16,777,215 possible test patterns when using random pattern generator.

4. The control logic for the On-Board ROM Test is simple when compared to the Random Test Pattern Generation method which requires loading seed patterns and special test sequencing.

5. Read control logic and address and data buses may possibly be shared between test and function purposes.

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM BIT

CATEGORY: LONG TUTORIAL — **PAGE** 3 **of** 14

SUBCATEGORY: BIT TECHNIQUE DISADVANTAGES

DATA TYPE: TEXT ☐ LIST [✓] TABLE ☐ GRAPHIC ☐ EQUATIONS ☐

DATA:

ON-BOARD ROM BIT
DISADVANTAGES

1. With the growing complexity of electronic circuitry being implemented on Line Replaceable Modules (LRM) of today, it is becoming more and more difficult for a test engineer to understand what he is testing, especially when under pressure to establish the test plan quickly. Without a true understanding of what is to be tested, it is nearly impossible to effectively and efficiently determine the test patterns that are necessary.

2. When the number of test patterns required to obtain adequate fault coverage is large and/or the number of CUT inputs is small or can be partitioned into a few small number of input groups, then the real estate required for the On-Board ROM Method becomes excessive when compared to the random pattern generation method.

3. Memory elements in general are not as reliable as random logic microelectronic devices.

4. Circuit design changes often require reprogramming the ROMs.

5. If the number of bus lines required to address the ROMs are excessive and/or the distance between the TP ROMs and the control logic, or between the GMR ROMs and the control logic is substantial, then Printed Circuit Board (PCB) real estate consumed is excessive and costly.

6. Memory allocated to either store test patterns or GMRs can never serve both test and function purposes as can shift registers used in Built in Logic Block Observers (BILBO) for example.

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM	
CATEGORY: LONG TUTORIAL	PAGE 4 of 14
SUBCATEGORY: BIT TECHNIQUE ATTRIBUTES	

DATA TYPE: TEXT ☐ LIST [✓ X] TABLE ☐ GRAPHIC ☐ EQUATIONS ☐

DATA:

ON-BOARD ROM BIT
ATTRIBUTES

1. REAL ESTATE PENALTY
   - Increases with CUT complexity
   - ROMs - Number of test patterns is approximately the cube of the number of gates for combinational. FFs increase the number even further
   - Control - Approximately 11 chips for this example. Number of counter chips increases with number of test patterns
   - Multiplexer - Number multiplexer chips equals number input lines divided by number of lines switched by multiplexer chip
   - Comparator - Number comparator chips equals (number output lines) divided by number of lines compared by chip
   - Land real estate depends on layout

2. POWER PENALTY
   - Roughly proportional to real estate penalty example:
     Power Penalty equals Percent Real Estate Penalty multiplied by CUT Normal power.
     - Exceptions (some ROMS have power down mode)
     - Switch Technology (use Metal Oxide Semiconductors (MOS) ROMS for higher density)

3. RELIABILITY PENALTY
   - Proportional to Real Estate Penalty if similar technology is used for Built in Test Equipment (BITE) as for CUT
   - May have to distinguish BITE failures that only effect BITE vs BITE failures that effect CUT
   - Computer Aided Design (CAD) System may have software package for reliability calculation

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM

CATEGORY: LONG TUTORIAL — PAGE 5 of 14

SUBCATEGORY: BIT TECHNIQUE ATTRIBUTES

DATA TYPE: TEXT ☐ LIST ☒ TABLE ☐ GRAPHIC ☐ EQUATIONS ☐

DATA:

ON-BOARD ROM BIT
ATTRIBUTES
(CONT)

4. TIMING PENALTY
   - Test Time Duration - Number of Test Patterns multiplied by Pattern Application Period
   - Circuit throughput Delay - Additional delays of Multiplexers
5. NON-CONCURRENT
6. CONCEPTUAL COMPLEXITY
   - Straight Forward
7. HARDWARE/SOFTWARE
   - Test Patterns in Firmware
8. TECHNOLOGY
   - All current digital technologies
   - May use higher density technologies for ROM to reduce real estate penalty. (May need MOS-Transistor Transistor Logic (TTL) converters)
9. IS BITE SELF TESTABLE?
   - Can do check sum on ROMs (add hardware)
   - Some ROMs have shadow registers
10. DESIGN COST
    - Use standard estimating procedures based on number of chips
    - Must add Engineering time to create Test Patterns and GMRs
    - May need debug time to hardware verify proper operation

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM	
CATEGORY: LONG TUTORIAL	PAGE 6 of 14
SUBCATEGORY: BIT TECHNIQUE ATTRIBUTES	

DATA TYPE: TEXT ☐ LIST ☒ TABLE ☐ GRAPHIC ☐ EQUATIONS ☐

DATA:

ON-BOARD BIT ROM
ATTRIBUTES
(CONT)

11. SOFTWARE DESIGN COST
    - Only applicable at system level

12. NUMBER OF BYTES OF STORAGE REQUIRED
    - Function of complexity of circuit (see Real Estate Penalty)

13. STAND-ALONE (SELF-CONTAINED BIT)?
    - Yes

14. WEIGHT
    - Proportional to real estate penalty weight
    - PENALTY = (Percent Real Estate Penalty) x (Weight of circuit)

15. Commercially available integrated circuits with testability features ROMs are available with shadow registers.

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM	
CATEGORY: LONG TUTORIAL	PAGE 7 of 14
SUBCATEGORY: DEFAULT DESIGN	

DATA TYPE: TEXT ☐ LIST ☒ TABLE ☐ GRAPHIC ☐ EQUATIONS ☐

DATA:

a) See figure 3 for ON-BOARD ROM LEVEL II BLOCK DIAGRAM.

b) See figure 4 for TEST PATTERN AND GOOD MACHINE RESPONSE ROM DEFAULT DESIGN.

c) See figure 5 for GOOD MACHINE RESPONSE COMPARISON LOGIC DEFAULT DESIGN.

d) See figure 6 for INPUT MULTIPLEXER DEFAULT DESIGN.

e) See figure 7 for CONTROL LOGIC FOR ON-BOARD ROM DEFAULT DESIGN.

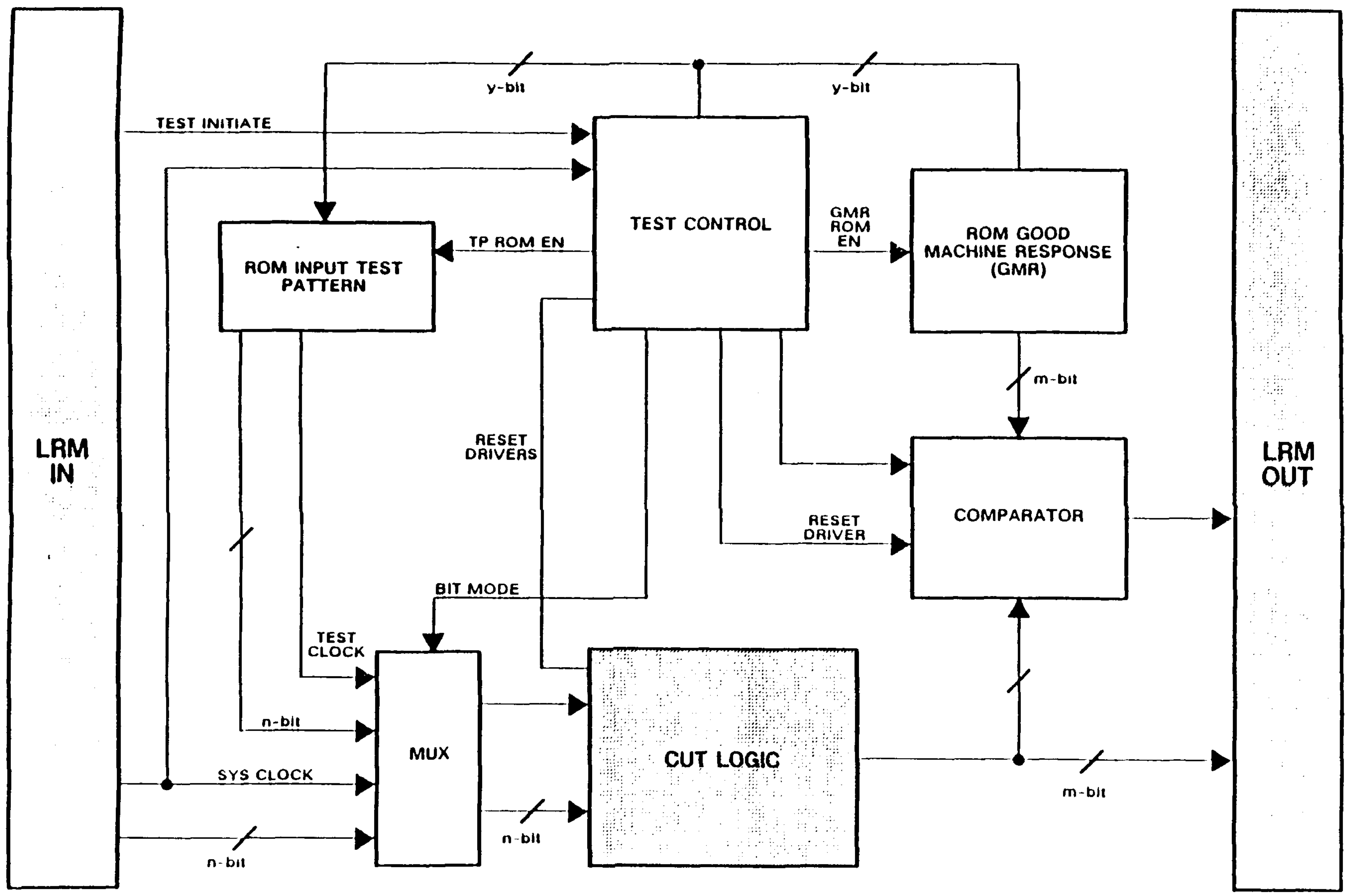


FIGURE 3 ON-BOARD ROM LEVEL II BLOCK DIAGRAM

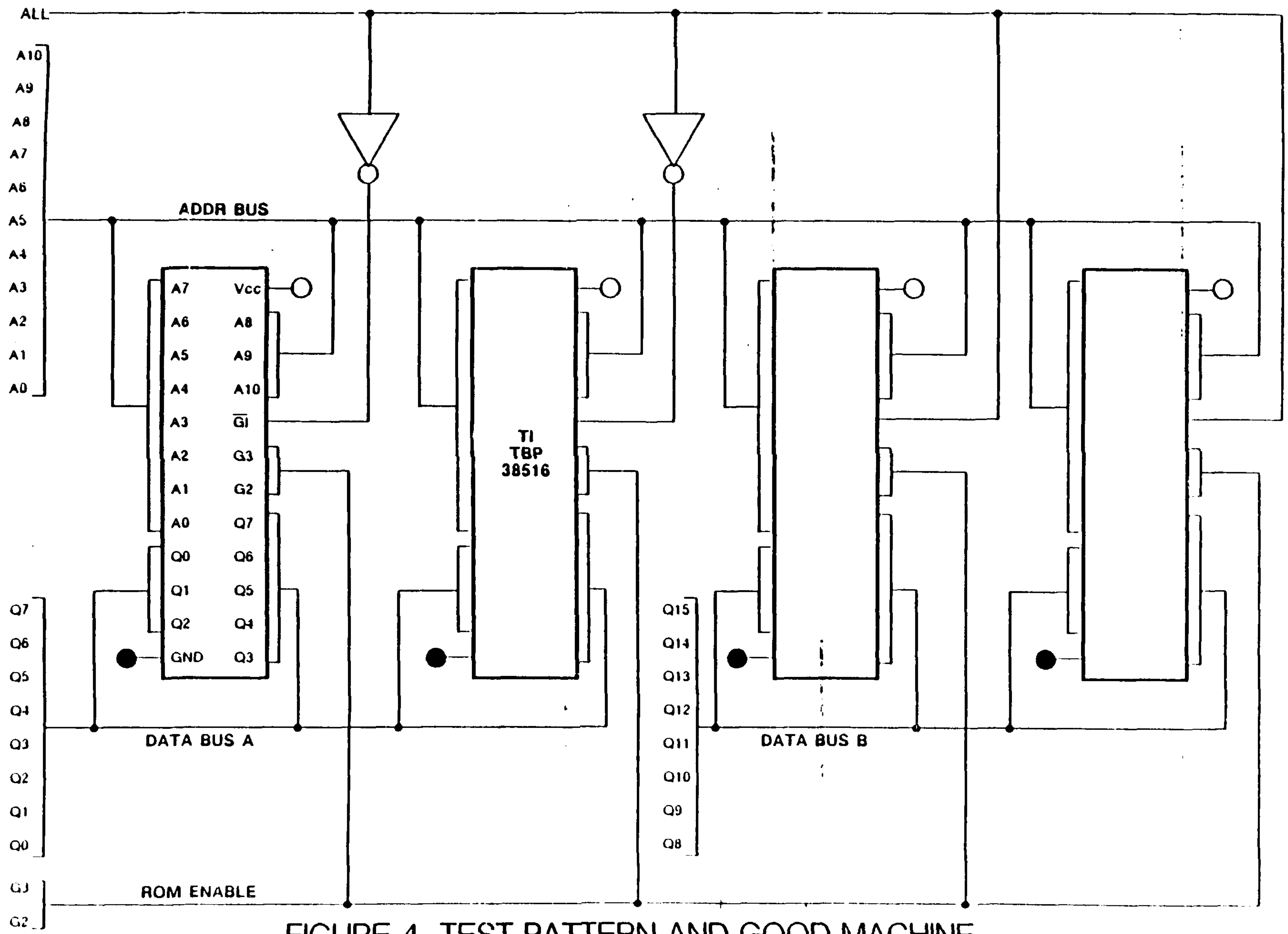


FIGURE 4 TEST PATTERN AND GOOD MACHINE RESPONSE ROM

PAGE 9 of 14

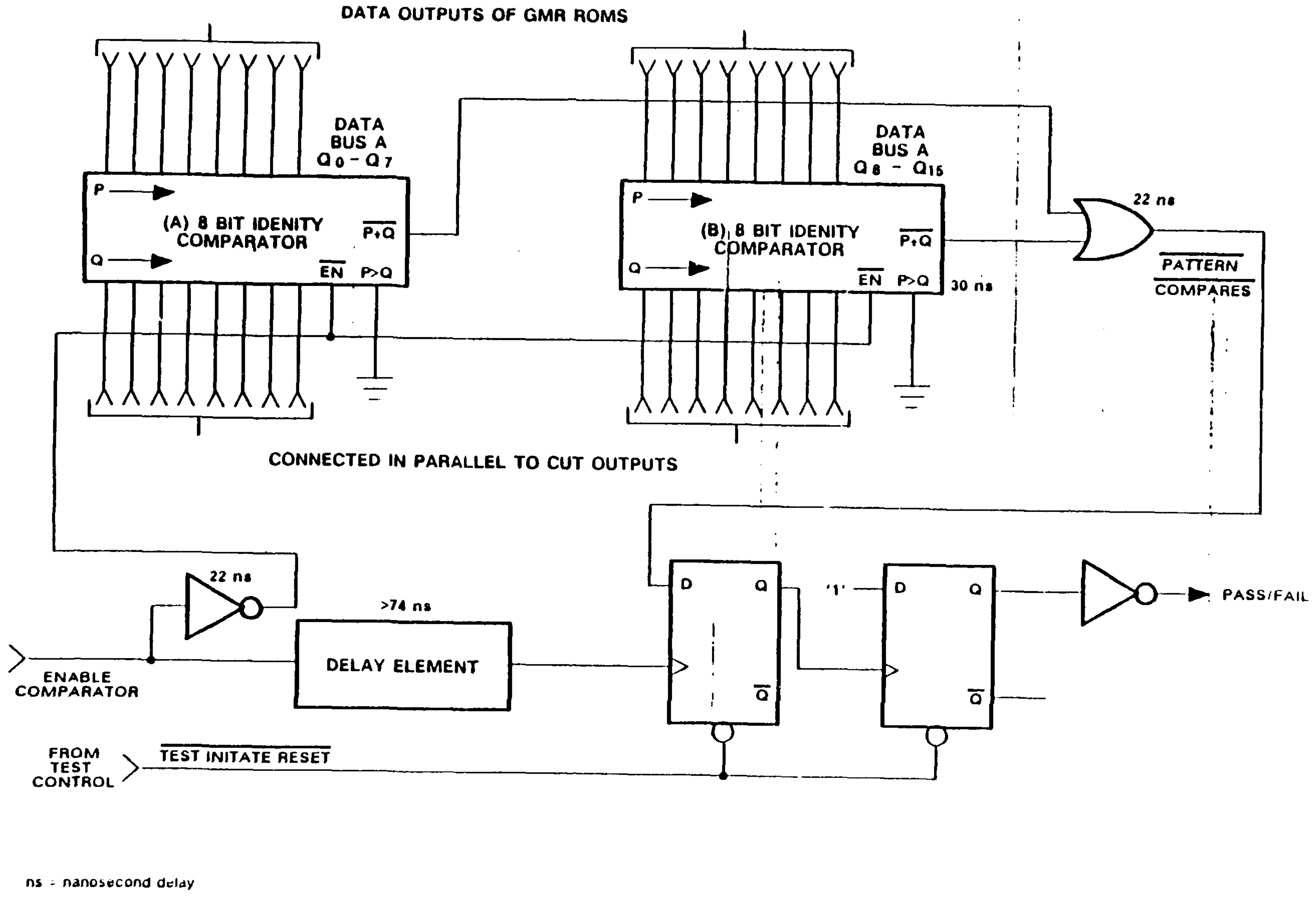
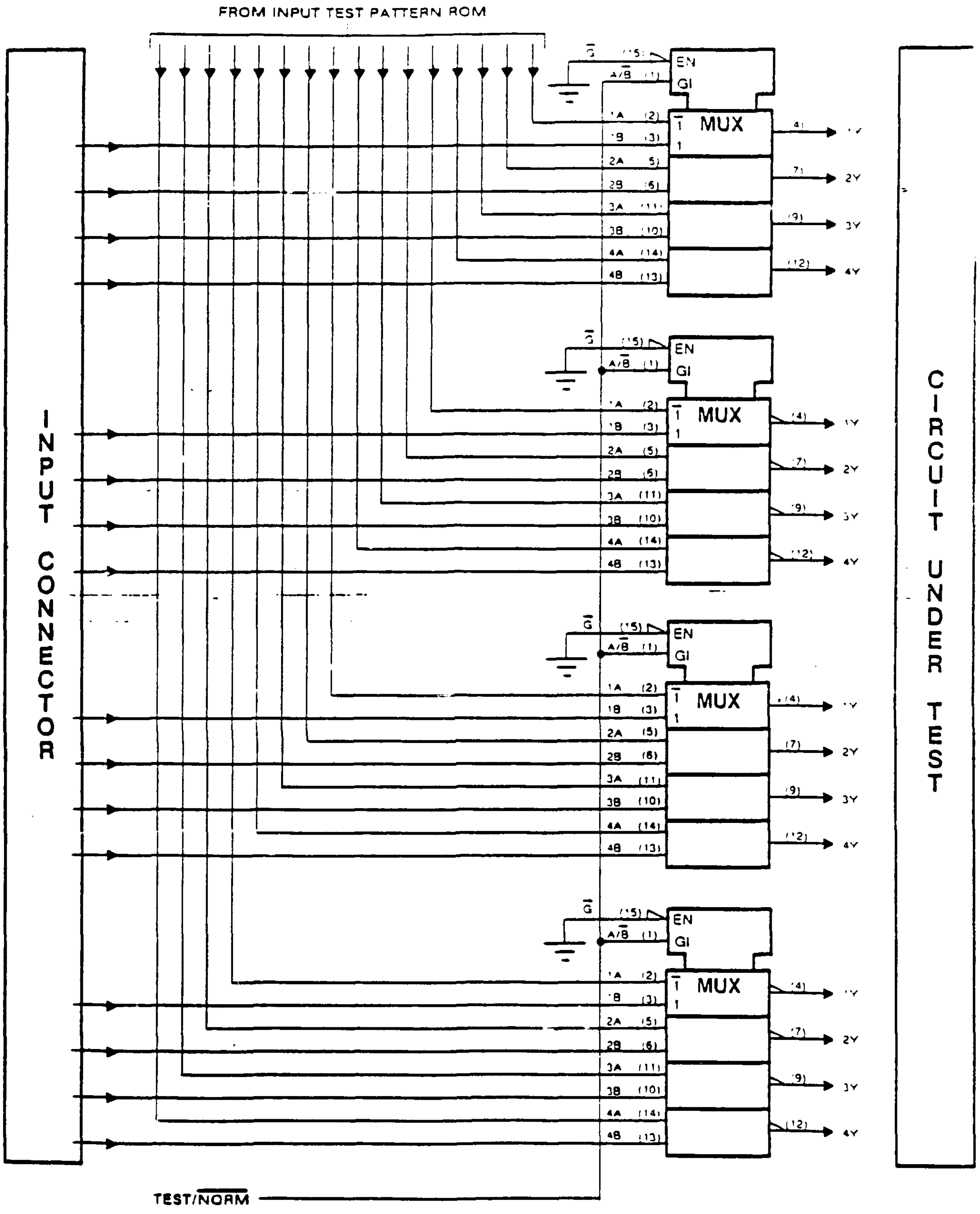


FIGURE 5 GOOD MACHINE RESPONSE COMPARISON LOGIC

PAGE 10 of 11



NOTE: 2 QUAD 2 TO 1 LINE DATA SELECTOR/MUX's CAN BE REPLACED BY AN OCTAL 2 INPUT MUXED LATCH-LS604.

FIGURE 6 INPUT MULTIPLEXER

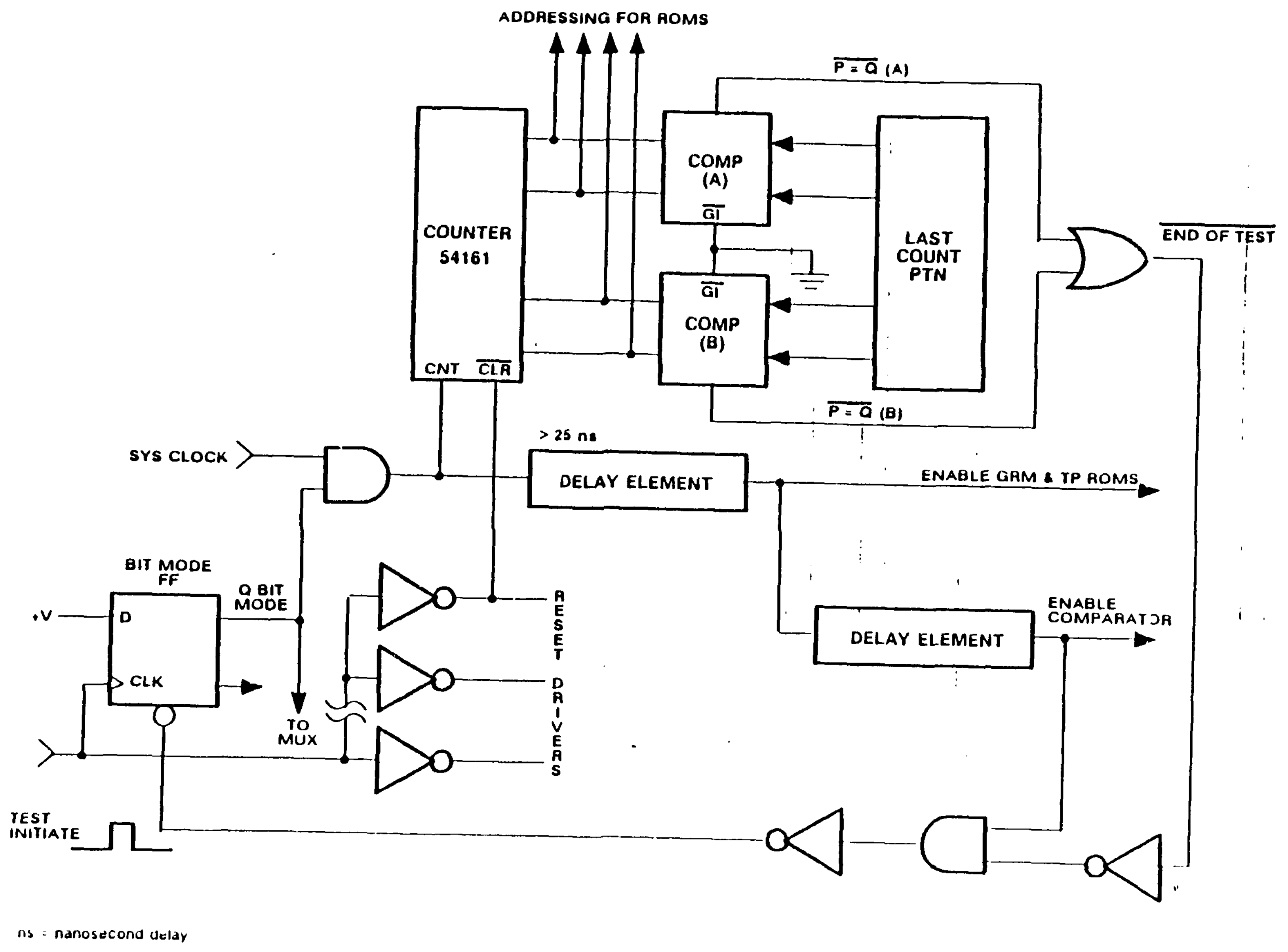


FIGURE 7 CONTROL LOGIC FOR ON-BOARD ROM

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE:	ON-BOARD ROM
CATEGORY:	LONG TUTORIAL — PAGE 13 of 14
SUBCATEGORY:	PART DATA TABLE

DATA TYPE: TEXT ☐ LIST ☐ TABLE ☒ GRAPHIC ☐ EQUATIONS ☐

DATA:

ON-BOARD ROM
PART DATA TABLE

NUMBER/NAME	AREA (sq in)	# OF PINS	POWER TYPICAL(mW)	POWER MAX. (mW)	WEIGHT (gms)
TBP385L16 2K x 8 PROM	0.375	24	325	500	6.5
74LS604/ OCT 2-IN MUXs LATCHES	0.87	28	275	350	7.5
74LS686/ 8 BIT MAG/ IDENT COMP	0.375	24	220	375	6.5
741617/ 4 BIT SYNC BIN COUNTER	0 243	16	315	455	2
7404/ HEX INVERTERS	0.243	14	90	165	2
7400/ QUAD 2-IN POS NAND	0.243	14	60	110	2
74125/ QUAD D FLIP FLOP	0.243	16	55	90	2

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM	
CATEGORY: LONG TUTORIAL	PAGE 14 of 14
SUBCATEGORY: BIBILOGRAPHY	

DATA TYPE: TEXT ☐ [✓ LIST ☒] TABLE ☐ GRAPHIC ☐ EQUATIONS ☐

DATA:

NONE REQUIRED

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM

CATEGORY: USER REQUESTED DATA PAGE of

SUBCATEGORY:

DATA TYPE: TEXT ☐ LIST ☒ TABLE ☐ GRAPHIC ☐ EQUATIONS ☐

DATA:

QUESTIONS	VARIABLE ASSIGNMENT
1. How many primary input pins are used by the PCB's operational circuitry?	v1
2. How many primary output pins are used by the PCB's operational circuitry?	v2
3. How many test patterns are required to be stored in the ROMs?	v3
4. What is the test pattern application rate?	v4
5. What is the estimated initialization time?	v5

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM

CATEGORY: EQUATIONS — **PAGE** 1 of 2

SUBCATEGORY: (DATA NOT TO BE DISPLAYED)

DATA TYPE: TEXT ☐ LIST ☐ TABLE ☐ GRAPHIC ☐ EQUATIONS ☒

DATA:

I) VARIABLE DEFINITIONS

n1= Number of ROM chips
n2= Number of MUX chips
n3= Number of COMPARATOR chips
n4= Number of COUNTER chips
n5= Number DECODE chips
n6= Number of PROGRAMMABLE DELAY chips
n7= BIT MODE status FF
n8= Number of CONTROL GATES
v1= Number of INPUT PINS<= 120
v2= Number of OUTPUT PINS<= 120
v3= Number of TEST PATTERNS<= 12288
v4= PATTERN RATE
v5= INITIALIZATION TIME

II) COMPONENT DETERMINATION EQUATIONS

$n1 = (v1/8)*(v3/2048) + (v2/8)*(v3/2048)$
$n2 = (v1/8)$
$n3 = (v2/8)$
$n4 = (v3/16)$
$n5 =$ Integer of $((n4 + 1)/2)$
$n6 = 2$
$n7 = 1$
$n8 = 2$

LIBRARY ELEMENT DATA SHEET

BIT TECHNIQUE: ON-BOARD ROM

CATEGORY: EQUATIONS — **PAGE** 2 of 2

SUBCATEGORY: (DATA NOT TO BE DISPLAYED)

DATA TYPE: TEXT ☐ LIST ☐ TABLE ☐ GRAPHIC ☐ EQUATIONS ☒

DATA:

III) PENALTY EQUATIONS

a) AREA (sq in)

Area of BIT chips = (.375)n1 + (.87)n2 + (.375)n3 + (.243)n4 + (.375)n5 + (.375)n6 + (.243)n7 + (.243)n8

Total area of BIT circuitry = (Area of BIT chips) + 15% for PC traces = 1.15 (Area of BIT chips)

b) POWER (mW)

Power = (325)n1 + (350)n2 + (375)n3 + (455)n4 + (375)n5 + (200)n6 + (90)n7 + (110)n8

c) WEIGHT (gms)

Weight of BIT chips (grams) = (6.5)n1 + (7.5)n2 + (6.5)n3 + (2.0)n4 + (6.5)n5 + (6.0)n6 + (2)n7 + (2)n8

Weight of BIT circuitry = Weight of BIT chips + 10% For Weight of solder = 1.1 (Weight of chips)

d) TIME (ns)

Test time = (v3) (v4) + v5

Throughput delay = 30

APPENDIX B
SELECTION RANKING ALGORITHM

APPENDIX-B

SELECTION RANKING ALGORITHM

-1- PURPOSE

This algorithm is used to generate an estimated penalty ranking of all BIT techniques which were considered suitable for a particular PCB design profile.

-2- OBTAIN RAW PENALTY DATA

Use the formulas for each penalty (from the BIT Technique Data Packages) to generate the "raw" penalty data. This data is represented by the "p" array where p_{ij} is the penalty for penalty parameter j of technique i.

-3- PENALTY ARRAY

From "raw" penalty data, create array "p". The array, p_{11}, for k penalty parameters and m suitable techniques is shown below:

$$\begin{array}{llcll} p_{11} & p_{12} & \dots & p_{1k} & \text{technique 1} \\ p_{21} & p_{22} & \dots & p_{2k} & \text{technique 2} \\ \cdot & & & \cdot & \\ \cdot & & & \cdot & \\ p_{m1} & p_{m2} & \dots & p_{mk} & \text{technique m} \end{array}$$

-4- COMPUTE AVERAGE PENALTY

Compute the average penalty P_j for each penalty parameter j of all m suitable techniques as follows :

$$P_j = \frac{1}{m} \sum_{i=1}^{i=m} p_{ij}$$

SELECTION RANKING ALGORITHM

-5- NORMALIZE P_{ij} ARRAY

Use the P_j ' s to normalize the p_{ij} array, resulting in the N_{ij} array where

$$N_{11} = \frac{p_{11}}{P_1} \quad \text{or} \quad N_{ij} = \frac{p_{ij}}{P_j}$$

The resulting N matrix is shown below :

$$\begin{array}{llll l} N_{11} & N_{12} & \dots & N_{1k} & \text{technique 1} \\ N_{21} & N_{22} & \dots & N_{2k} & \text{technique 2} \\ \cdot & & & \cdot & \\ \cdot & & & \cdot & \\ N_{m1} & N_{m2} & \dots & N_{mk} & \text{technique m} \end{array}$$

-6- APPLY PENALTY WEIGHTING FACTORS

For each penalty parameter (area, power, etc.), there also exists a corresponding weighting factor W. These are user supplied during the User Profile Generation or the default values are used. The weighting factors make up a single dimension weighting factor, **W**. Applying **W** to the N array as follows

$$W_{ij} = W_j \bullet N_{ij}$$

creates a normalized weighted array, W, with the array elements defined as follows :

$$\begin{array}{llll l} W_{11} & W_{12} & \dots & W_{1k} & \text{technique 1} \\ W_{21} & W_{22} & \dots & W_{2k} & \text{technique 2} \\ \cdot & & & \cdot & \\ \cdot & & & \cdot & \\ W_{m1} & W_{m2} & \dots & W_{mk} & \text{technique m} \end{array}$$

SELECTION RANKING ALGORITHM

-7- DETERMINE BIT TECHNIQUE RANKING

R is the column Ranking Array. The selection ranking R_i for technique i is then

$$R_i = \sum_{j=1}^{j=k} W_{ij}$$

Technique ranking will cause those techniques with a lower overall penalty ranking to be sorted to the top. It is expected that techniques with similar overall penalty values may not clearly define the better ranking as superior. In these cases, engineering judgement and a closer inspection of penalty weighting factors may help discriminate between the closely ranked techniques.

****

****

APPENDIX C
EVALUATION ALGORITHM

APPENDIX-C

EVALUATION ALGORITHM

-1- PURPOSE

This algorithm is used to generate an actual penalty ranking of all the circuits in a specified PCB.

-2- APPLY BIT FACTORS

For each circuit in *data/cadbit-eval-data*, apply the BIT factor b_i to the corresponding circuit penalties q_{ij} in data/ppd where j represents the penalty parameters of circuit i. Using the following formula :

$$P_{ij} = b_i * q_{ij} \quad (\text{ where } i = 1 \text{ to m and } j = 1 \text{ to n });$$

results in a penalty array P_{ij} where the penalties have a weighting factor equal to the BIT factor.

$$\begin{array}{llcll} P_{11} & P_{12} & \ldots & P_{1n} & \text{circuit 1} \\ P_{21} & P_{22} & \ldots & P_{2n} & \text{circuit 2} \\ \cdot & & & \cdot & \\ \cdot & & & \cdot & \\ \cdot & & & \cdot & \\ P_{m1} & P_{m2} & \ldots & P_{mn} & \text{circuit m} \end{array}$$

-3- OBTAIN BIT PENALTIES BY BIT GROUP

For every BIT group k sum each of the penalty parameters P_{ij} resulting in the B_{kj} array where B_{kj} is the actual value of penalty parameter j of BIT group k. The following formula :

$$B_{kj} = \sum_{i=1}^{l=m} P_{ij} * A$$

where $A = 0$ if the BIT Group for circuit i is not equal to k and where $A = 1$ if the BIT Group for circuit i is equal to k, this results in :

$$\begin{array}{lllll}
B_{11} & B_{12} & \cdots & B_{1n} & \text{BIT Group 1} \\
B_{21} & B_{22} & \cdots & B_{2n} & \text{BIT Group 2} \\
\vdots & & & \vdots & \\
B_{m1} & B_{m2} & \cdots & B_{mn} & \text{BIT Group m}
\end{array}$$